Universidade de Aveiro
2008
Departamento de Electrónica Telecomunicações e Informática

CARLOS FILIPE MARQUES ALMEIDA

ANONIMATO EM SISTEMAS DE VOTAÇÃO ELECTRÓNICA

Universidade de Aveiro
2008
Departamento de Electrónica Telecomunicações e Informática

CARLOS FILIPE MARQUES ALMEIDA

ANONIMATO EM SISTEMAS DE VOTAÇÃO ELECTRÓNICA

dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Professor Doutor André Ventura da Cruz Marnôto Zúquete, Professor Auxiliar do Departamento de Electrónica Telecomunicações e Informática da Universidade de Aveiro

Apoio financeiro da FCT e do FSE no âmbito do III Quadro Comunitário de Apoio.

Dedico este trabalho aos meus pais e à Sara por todo o apoio e compreensão.

o júri

presidente

Prof. Dr. José Luís Guimarães Oliveira
professor associado da Universidade de Aveiro

Prof. Dr. Luís Eduardo Teixeira Rodrigues
professor catedrático do Departamento de Engenharia Informática do Instituto Superior Técnico da Universidade Técnica de Lisboa

Prof. Dr. André Ventura da Cruz Marnôto Zúquete
professor auxiliar da Universidade de Aveiro

agradecimentos

Ao Professor André Zúquete por todo o seu apoio, em especial pelas estimulantes reuniões de trabalho que em muito ajudaram na evolução deste trabalho, bem como pela compreensão demonstrada.

palavras-chave Voto electrónico, anonimização, tolerância a falhas, Mix Rings, REVS

resumo Este trabalho descreve a concepção e realização de uma nova arquitectura para a submissão anónima de votos concebida para o sistema de votação electrónica REVS. A arquitectura é baseada no conceito dos Mix Rings, um anel de entidades que permite trocas de dados anónimas. O conceito original de anonimização de tráfego dos Mix Rings foi mantido mas a nova arquitectura, concebida para o problema concreto da submissão de votos, é diferente em múltiplos aspectos, a saber: política de construção do anel, gestão dinâmica do anel para tolerar a falha e recuperação dos seus elementos e autenticação dos elementos do anel. Um protótipo do anel de anonimização foi implementado em Java, usando Java RMI como base para a interacção remota, e foi testado em múltiplos cenários operacionais envolvendo falhas singulares ou múltiplas.

keywords e-Voting, anonimyzation, fault tolerance, Mix Rings, REVS

abstract This work describes the design and implementation of a new architecture for anonymous ballot submissions within REVS (Robust Electronic Voting System). The architecture is based on the Mix Rings concept, which was conceived for supporting anonymous unidirectional communications between two hosts. However, there are many differences between the original Mix Ring architecture and the one here presented; some of them deserve special attention on this work: ring construction and management for fault tolerance, which is critical in voting events, and mutual authentication of Mixes participating in the ring. To validate our architecture we implemented a prototype in Java. The communication of voters with the ring and within Mixes of the ring is implemented on top of Java RMI. Several fault scenarios were conceived and tested to assess the robustness against singular or multiple Mix failures.

Índice Geral

Índice de Figuras

Capítulo 1

Introdução

Com a massificação da utilização da Internet cresce também o potencial uso dos sistemas de votação electrónica sobre esta rede. O crescimento da utilização destes sistemas de *e-voting* será proporcional à confiança que os utilizadores depositem nos mesmos. Essa confiança será maior se for possível a verificação, pelo votante, de que o processo de submissão do seu voto decorreu de forma correcta. Contudo, qualquer informação enviada como retorno para o utilizador não pode colocar em causa a privacidade inerente ao acto eleitoral ou mesmo requisitos de não-coacção [1].

Uma forma possível de alcançar esse objectivo é através dos Mix Rings [2]. Estes combinam princípios de operação das Mix Nets [3] e do Onion routing [4] para providenciar anonimato às comunicações *peer-to-peer* (P2P) através de uma topologia lógica em anel, sendo este constituído por diversas entidades que colaboram de modo a propiciar o anonimato da comunicação extremo-a-extremo. Esta solução representa um compromisso entre o anonimato fraco e a baixa latência do Onion routing e o anonimato forte e a alta latência das Mix Nets.

O Mix Ring mantém um ritmo constante de troca de mensagens entre Mixes para esconder o tráfego útil, conseguido através da submissão das mensagens reais intercaladas com mensagens de camuflagem (*cover traffic*).

Todas estas mensagens são construídas com recurso a chaves de cifra, negociadas previamente, sendo as cifras realizadas pela ordem inversa de recepção. Uma vez que apenas o iniciador do anel conhece todos os participantes no sistema e respectiva ordenação, torna-se necessário que para cada camada seja adicionada à mensagem informação relativa ao próximo nó no anel, identificando o Mix seguinte e para o qual a mensagem deverá ser reenviada. Por outras palavras, o encaminhamento da mensagem é ditado pela própria a cada Mix do anel. À medida que a mensagem vai progredindo no anel, as sucessivas camadas vão sendo removidas, por decifra, até retornar ao iniciador do anel. Deste modo, podemos ver as mensagens de *cover traffic* como mensagens com conteúdo desprezável que vão sendo reenviadas de Mix para Mix ao longo do anel, sendo o último destinatário o próprio emissor. No entanto, para as mensagens com informação relevante é escolhido um Mix aleatoriamente e utilizado um mecanismo de *fan-out*, o qual determina a divisão da mensagem recebida. Assim, nesse Mix uma parte da mensagem, correspondente a *cover traffic*, é enviada para o próximo Mix e outra parte para o receptor identificado na mensagem. Na Figura 1, pode ser visualizada uma representação do processo de comunicação através dos Mix Rings.

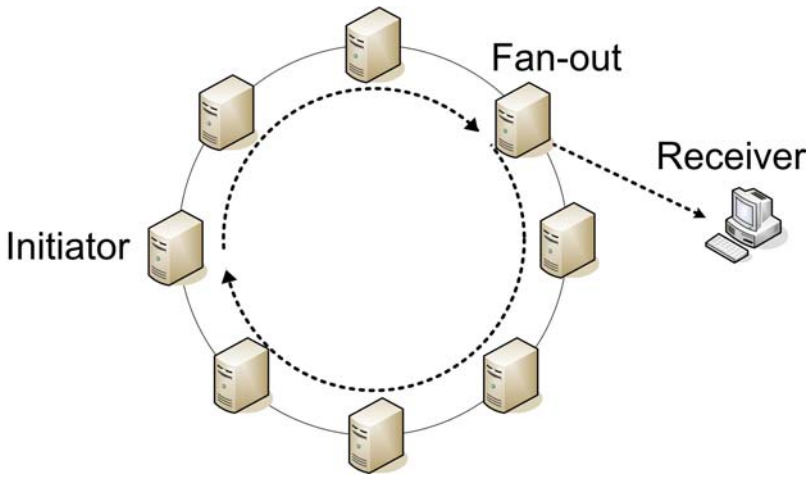

Figura 1 – Arquitectura de um Mix Ring, cujo objectivo é permitir a comunicação anónima entre o Initiator e o Receiver

Nos Mix Rings não existe uma visão global do anel por todos os Mixes. O anel é criado por um dos Mixes e os demais ficam a conhecer apenas o seu Mix seguinte, o qual vem explicitamente referido nas mensagens que recebem. Assim, um Mix Ring não recupera facilmente de uma falha ocorrida num dos Mixes, uma vez que o seu antecessor

no anel não possui mais nenhuma informação sobre o mesmo. Assim, na presença de uma falha no anel, a mesma é detectada pelo iniciador do anel através quebra na recepção das mensagens de *cover traffic*, que podem assim ser vistas analogamente como o batimento cardíaco do anel. A recuperação implica que o Mix iniciador do anel realize novamente todo o processo de construção de um novo anel.

Adaptando este conceito à deposição anónima de votos num sistema de *e-voting*, um Mix Ring pode ser usado para receber os votos submetidos pelos votantes, depositar o voto num dos elementos do anel e proceder à devolução de um recibo anónimo, preparado pelo votante mas sem qualquer relação com o conteúdo do seu voto [5].

Esta arquitectura de submissão anónima de votos com recibo foi originalmente concebida para o sistema REVS (*Robust Electronic Voting System* [6]). No entanto, apesar do REVS ter servido como ponto de partida, esta arquitectura não se limita ao mesmo, podendo ser utilizada em qualquer sistema que necessite de um canal de comunicação seguro e anónimo entre os extremos envolvidos na comunicação. Nomeadamente, pode ser usada em sistemas de *e-voting* similares, i.e., baseados em assinaturas às cegas (*blind signatures*).

No REVS a comunicação anónima entre o votante e o servidor responsável pelo armazenamento dos boletins de voto é conceptualmente realizada por servidores designados por Anonymizers. Estes têm como principal função impedir a correlação entre as máquinas onde é executada a aplicação cliente (usada pelo votante) e os Counters, de modo a evitar o emparelhamento entre votos e votantes. Os Anonymizers podem ser realizados de diversas formas, por exemplo, recorrendo a Mix Nets [3] ou a Onion Routing [4]. No entanto, tal significa um aumento significativo do número de servidores envolvidos no processo: múltiplos servidores envolvidos na concretização de cada Anonymizer e múltiplos Counters para tolerância a falhas do processo de deposição do voto.

Para resolver este problema optou-se por combinar os dois tipos de entidades, Anonymizers e Counters, fundindo-os numa espécie de Mix Ring (ver Figura 2). Deste modo, a função de estabelecimento do canal anónimo passa a ser desempenhada pelos servidores Counter, que formam um anel lógico com uma funcionalidade próxima de um Mix Ring. O votante envia o seu voto para um ***Submission Counter*** e o anel de Counters usa a funcionalidade de *fan-out* para deposição do voto num ***Storage Counter*** (uma

operação de *fan-out* interna ao anel) e para envio do recibo ao votante por um ***Receipt Counter*** (uma operação de *fan-out* externo). Os Counters que operam como *Submission Counter*, *Storage Counter* e *Receipt Counter* são escolhidos pelo votante, bem como os demais Counters que irão processar o seu voto ao longo da circulação pelo anel. Naturalmente, para garantir o anonimato do votante os três Counters que lidam com informação sua – o voto submetido, o voto guardado e o recibo enviado – não devem conseguir relacioná-la.

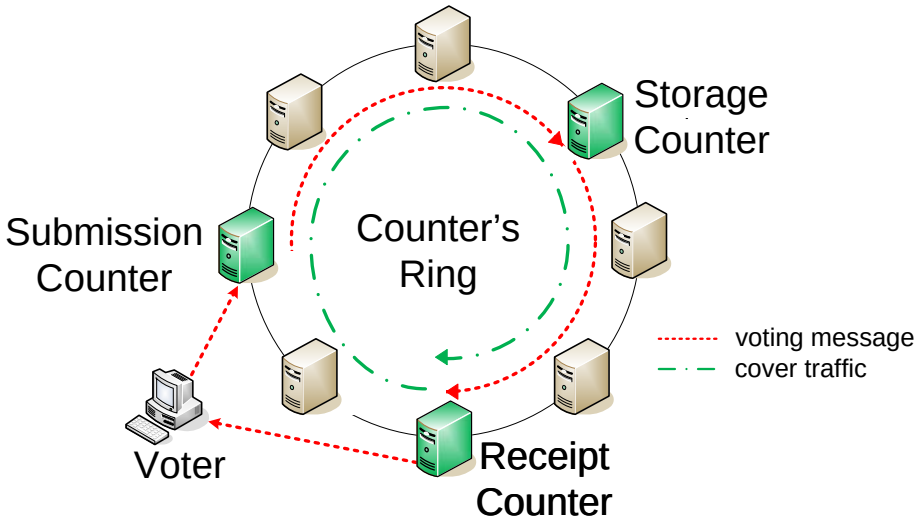

Figura 2 - Nova arquitectura para anonimato e submissão de votos proposta para o REVS, baseada no conceito de Mix Rings.

1.1. Objectivos

O anonimato, quase sempre associado ao direito à privacidade do votante, desempenha um papel preponderante na definição dos mais variados cenários de eleições. No entanto, a curiosidade e desconfiança, intrínseca ao ser humano, aliadas à falta de informação durante o desenrolar do processo de votação conduzem à quebra de confiança dos utilizadores nestes sistemas electrónicos.

Por analogia com os sistemas de votação tradicionais, presenciais e com boletim de voto em papel, facilmente se compreende que a confiança do votante reside no facto de o próprio verificar a inclusão do seu voto na urna da eleição para posterior contagem. Deste modo, os sistemas de votação electrónica deverão acompanhar essas práticas e, sem

comprometer o direito à privacidade do eleitor, fornecer uma informação credível de que o processo de submissão ocorreu sem problemas. Este *feedback* ao utilizador é conseguido através da adopção de uma arquitectura baseada nos Mix Rings para submissão anónima dos votos.

Contudo, de nada vale aumentar esta confiança dos utilizadores se o próprio sistema não possuir a robustez necessária para garantir a sua disponibilidade ao longo de toda a eleição. Isto conduz à necessidade de se olhar não só para o anonimato das comunicações como também para os requisitos de tolerância a falhas do sistema [7]. Assim, deve-se garantir que o sistema funcione mesmo que ocorram ausências por parte das entidades constituintes da rede. Do mesmo modo, deve-se projectar o sistema para recuperar de falhas em tempo real, nomeadamente durante o intervalo (muito limitado) em que decorre um processo eleitoral, e manter o serviço disponível com os recursos remanescentes.

Para além da robustez em termos de tolerância a falhas, o sistema deverá de igual modo possuir robustez no que concerne a ataques em conluio por parte dos elementos constituintes do sistema. Através da autenticação e imposição de protocolos rígidos nos processos de construção e operação do anel, consegue-se garantir que apenas os elementos autorizados podem interagir no sistema, bem como reduz-se a possibilidade de cooperação entre os mesmos com o objectivo de quebrar o sigilo das comunicações.

O trabalho realizado, e descrito nesta dissertação, teve os seguintes objectivos:

- Estudo de protocolos de comunicação suportados por redes de alto nível (*overlay networks*) em anel e avaliação dos seus procedimentos para a construção, gestão e monitorização do funcionamento da rede;
- Proposta de modelação da arquitectura para submissão anónima de votos, compatível com o sistema REVS, baseada na arquitectura Mix Ring e tendo em conta as diferenças operacionais de ambos os sistemas e requisitos específicos definidos para o sistema alvo;
- Proposta de definição de um modelo para construção e gestão de mensagens análoga ao existente para o sistema Mix Ring. No entanto, os aspectos relacionados com a criação e processamento dos votos não foram extensivamente abordados;

- Estudo de soluções para a criação de um canal seguro com autenticação mútua sobre RMI;
- Desenvolvimento e implementação de um protótipo que possibilite o teste e validação do modelo proposto.

1.2. Organização do texto

Com o objectivo de sustentar a apresentação do trabalho desenvolvido ao longo desta dissertação, este documento encontra-se dividido e organizado do seguinte modo.

No Capítulo 2 será efectuado o enquadramento deste trabalho e apresentada a ideia que serviu de motivação à realização deste trabalho. Serão aqui abordados aspectos funcionais relacionados com o sistema de votação electrónica utilizado evidenciando as diferenças relativamente aos Mix Rings, as quais irão corresponder a alterações na definição do modelo da implementação.

No Capítulo 3 será apresentado, comparativamente, um resumo do trabalho de pesquisa sobre soluções existentes para comunicações em redes *overlay* com topologias baseadas em anel, independentemente do objecto principal do sistema.

Os requisitos para o desenvolvimento da nova arquitectura serão apresentados no Capítulo 4 , junto com alguns dos compromissos assumidos aquando do início da definição do novo modelo.

No Capítulo 5 será apresentada a definição da rede de suporte ao sistema, com a definição dos protocolos de construção, gestão e manutenção do anel de comunicações.

No Capítulo 6 será abordado o modelo para construção das mensagens contendo votos, bem como a estrutura desenvolvida para as mensagens de *cover traffic*. Neste capítulo será de igual modo apresentado o modelo de optimização do método de submissão de votos.

A implementação e respectiva discussão serão apresentadas no Capítulo 7 , sendo abordada a metodologia utilizada e apresentada a funcionalidade e interoperabilidade entre os diversos componentes do sistema.

Finalmente, no Capítulo 8 serão apresentadas as conclusões relativas ao trabalho desenvolvido e anunciadas propostas de trabalho futuro.

Capítulo 2

Enquadramento e motivação

O trabalho apresentado ao longo deste documento baseia-se no *Robust Electronic Voting System* (REVS) [6] um sistema de votação electrónica baseado em assinaturas às cegas (*blind signatures*), possibilitando ao eleitor a submissão de várias cópias do voto e consequente detecção de réplicas, por parte das entidades eleitorais, aquando da contagem final. Assim, apesar de esta implementação assentar no sistema REVS, a mesma será válida para outros sistemas semelhantes baseados em assinaturas cegas.

2.1. REVS – Robust Electronic Voting System

2.1.1. Arquitectura

O REVS usa 5 tipos de servidores que suportam várias eleições em simultâneo: *Commissioner*, *Ballot Distributor*, *Administrator*, *Anonymizer* e *Counter*. Existe ainda uma componente, o *módulo cliente*, que é usada pelos votantes para participarem em eleições e que esconde as complexidades do protocolo (por exemplo, a obtenção de assinaturas cegas).

O *Commissioner* existe para tratar receber queixas de votantes ou servidores. Para além disso, é ainda responsável por preparar eleições, gerar e manter secretas as chaves das eleições, assinar o boletim de voto para garantir a sua correcção e definir os parâmetros operacionais da eleição (endereços e chaves públicas dos servidores, número *t* de assinaturas requeridas, etc.).

O *Ballot Distributor* distribui dados, preparados e assinados pelo *Commissioner*, pelos votantes: eleições activas e respectivos boletins de voto e parâmetros operacionais. Usou-se um servidor dedicado, que pode ser livremente replicado para melhorar o desempenho e a disponibilidade, por ser uma operação de transferência de dados potencialmente intensa.

Os *N Administrators* garantem a democracia. Um voto só é válido para a contagem final se possuir um número mínimo de *t* assinaturas de diferentes *Administrators* e as assinaturas só são concedidas uma vez a cada votante autorizado. Se $t > N/2$ então cada votante só pode gerar um voto válido.

Os *Anonymizers* garantem o anonimato dos computadores dos votantes perante os *Counters* e baralham a ordem dos votos recebidos e retransmitidos para impedir análises temporais. Cada votante pode usar um número arbitrário de *Anonymizers*.

Os *Counters* acumulam e validam votos submetidos pelos votantes através dos *Anonymizers*. Para a contagem final são agrupados os votos de todos os *Counters* e eliminados os votos repetidos (cada voto possui um *bit commitment*, i.e. um identificador único gerado aleatoriamente pelo votante). Tal como no EVOX-MA, qualquer entidade com acesso aos votos acumulados pelos *Counters* pode efectuar a contagem final, o que é desejável para aumentar a confiança no resultado eleitoral.

Os votos entregues aos *Counters* vão cifrados com uma chave pública da eleição, gerada pelo *Commissioner*. Só após o fim da eleição é que a correspondente chave privada é divulgada de modo a permitir a contagem final. Caso seja desejável, os votos podem ser entregues em claro aos *Counters*, o que permite um apuramento em tempo real dos resultados eleitorais. Haverá, decerto, certas eleições onde tal poderá ter interesse.

O REVS suporta configurações sem pontos singulares de falha, uma vez que usa *N Administrators* e permite uma replicação livre dos demais servidores. O sistema tolera

também até *N-t* falhas de *Administrators*; se num dado instante o número de falhas for superior o sistema fica temporariamente incapaz de gerar votos válidos.

2.1.2. Protocolo de votação

O processo eleitoral é constituído por três fases: a preparação da eleição (criação de cadernos eleitorais, recolha dos elementos de identificação dos votantes autorizados, etc.), a realização da eleição, onde os eleitores expressam a sua opinião, e apuramento dos resultados.

A realização da eleição possui três grandes passos, geridos pelo *módulo cliente* (ver Figura 3), sendo o quarto uma operação interna do sistema correspondendo à entrega dos votos aos Counters por parte dos Anonymizers:

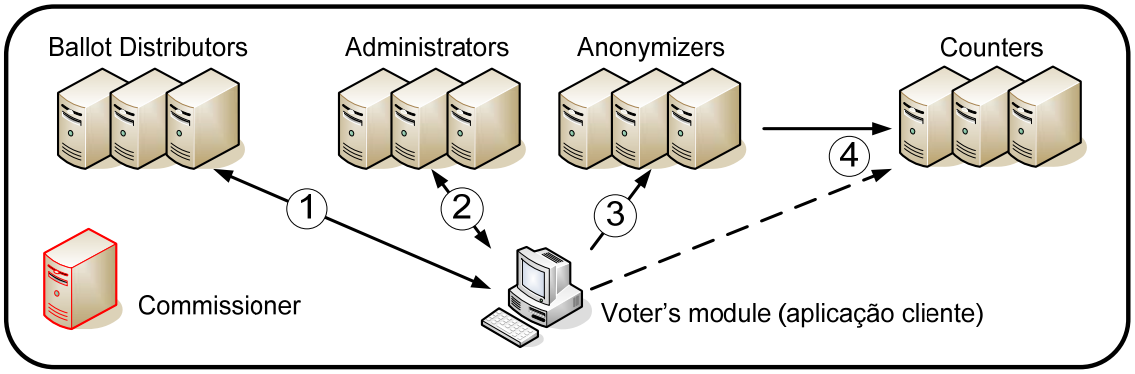

Figura 3 – Arquitectura original do sistema REVS

1 - Distribuição de boletins. O votante contacta um dos *Ballot Distributors* para pedir os elementos relativos a uma eleição -- boletim, chave pública e configuração operacional -- que são devolvidos assinados pelo *Commissioner*. Tal acontece em duas fases: primeiro o votante fornece a sua identificação e recebe a lista de eleições em que pode participar; em seguida pede os elementos relativos a uma dessas eleições.

2 - Criação de um voto válido. O votante expressa a sua opinião no boletim obtido e "entrega-o" para ser assinado cegamente pelos *Administrators*. Antes da primeira assinatura o *módulo cliente* cria e acrescenta ao boletim um identificador único, após o que envia o resumo (*digest*) do boletim, ocultado, para *t Administrators* para ser assinado. Antes desse envio o votante pode, e deve, guardar o estado do boletim num suporte não

volátil para permitir a repetição do processo em caso de falha (porque a repetição implica a reutilização do mesmo voto, do seu identificador único e dos factores de ocultação usados). Esta salvaguarda, dependendo de como o sistema for usado, pode ter implicações com garantias de não coacção.

Quando um *Administrator* recebe um pedido de assinatura verifica se o mesmo vem de um votante autorizado, autentica-o e devolve a assinatura requerida do resumo ocultado. Essa assinatura é guardada para mais tarde ser devolvida caso o mesmo votante torne a requerê-la (o que pode acontecer caso tenham ocorrido falhas), o que é relevante para garantir democracia.

As assinaturas recebidas pelo votante são transformadas, anulando o efeito da ocultação, revelando uma assinatura convencional de um *Administrator* sobre o resumo original do boletim de voto. Esta assinatura final pode ser validada por qualquer entidade, inclusive o votante, mas o *Administrator* que lhe deu origem não consegue relacioná-la com qualquer das assinaturas que produziu sobre resumos ocultados (o que garante correcção e privacidade).

Se o votante receber assinaturas inválidas tal pode impedi-lo de produzir um voto válido, devendo reclamar junto do *Commissioner*. São suficientes *N-t+1 Administrators* em conluio para impedir alguém de votar devolvendo assinaturas inválidas. Este problema pode ser minorado baixando tanto quanto possível o valor de *t*. Se *t* for igual a N/2+1 então o grau de conluio necessário para fabricar ou negar votos é igual.

3 - Entrega do voto. Depois de obtidas *t* assinaturas o voto está válido e pode ser entregue aos *Counters* através dos *Anonymizers*. O voto é constituído pelo boletim preenchido e pelas assinaturas dos *Administrators*, tudo cifrado com a chave pública da eleição e cifra híbrida,. Este voto não pode ser inspeccionado antes da eleição terminar, porque está cifrado, nem pode ser alterado sem que tal invalide todas as assinaturas, o que o tornaria automaticamente inválido. Assim, muito embora os *Anonymizers* e os *Counters* possam "perder" votos, não o podem fazer de maneira a controlar o resultado final da eleição. Por outro lado, o votante pode enviar uma cópia do seu voto para diversos *Counters*, pelo que a "perda" de uma dessas cópias pode ser facilmente tolerada.

O apuramento dos resultados só acontece depois de terminado o prazo estipulado para a eleição. Para esse apuramento o *Commissioner* revela a chave pública da eleição,

que permite verificar todos os votos entregues a todos os *Counters*. Todas as entidades com acesso à totalidade dos votos nos *Counters* podem, então, proceder ao apuramento de resultados. Nesse apuramento devem ser usados apenas os votos não repetidos com *t* assinaturas válidas de *Administrators*. Qualquer votante pode realizar esse apuramento e verificar se o seu voto, que possui um identificador único copiado para um dispositivo de salvaguarda no passo 2, foi ou não incluído no mesmo. Se o voto tiver sido perdido o votante pode enviá-lo anonimamente ao *Commissioner* para que este o inclua na contagem final.

Para terminar os *Administrators* publicam a lista de todas as assinaturas que produziram e para quem as produziram. Esta lista permite verificar se o conjunto de votos considerados no apuramento final é superior ao número de pessoas que efectivamente votaram, o que, a acontecer, revela uma fraude.

2.2. REVS - Nova arquitectura

A arquitectura original do REVS apresenta alguns problemas, tanto ao nível da definição de processos, como ao nível da segurança global do sistema e à confiança dos utilizadores. Apesar de assentar numa ideia relativamente simples e funcional, o processo de construção do canal de comunicação anónimo nunca foi especificado na sua plenitude. Do mesmo modo, a sua simplicidade introduz problemas ao nível da segurança, sendo necessário assumir que não existe conluio entre o Anonymizer e o Counter para alcançar a validação do modelo no que toca à privacidade. Por outro lado, a confiança dos utilizadores neste tipo de sistemas apenas poderá ser incrementada se o próprio sistema fornecer alguma informação ao utilizador sobre o estado de conclusão do processo, obviamente sem comprometer a privacidade do votante nem a sua resistência à coacção.

Para colmatar as deficiências existentes na fase de submissão anónima de votos surgiu uma nova proposta para a construção do canal de comunicação anónimo [5], baseada no conceito dos Mix Rings [2].

Os Mix Rings, ver Figura 4, inicialmente propostos para esconder os interlocutores numa comunicação ponto-a-ponto, representam uma solução de compromisso entre as Mix Nets [3] e as redes de Onion Rounting [4]. As mensagens são construídas por camadas, correspondendo cada camada à cifra do conteúdo com a chave pública do respectivo nó.

Com a inserção de mensagens de camuflagem, em tudo semelhantes às que contêm a informação relevante, consegue-se esconder o tráfego útil no âmbito do tráfego constante trocado em todo o anel. Para a comunicação com o receptor é utilizado um mecanismo de *fan-out*. Esta operação consiste na divisão da mensagem recebida, por um dado nó, em duas, sendo parte da mensagem encaminhada para o receptor pretendido e o resto da mensagem, contendo uma mensagem encoberta, reenviado para o anel de modo a manter o tráfego constante e inalterado.

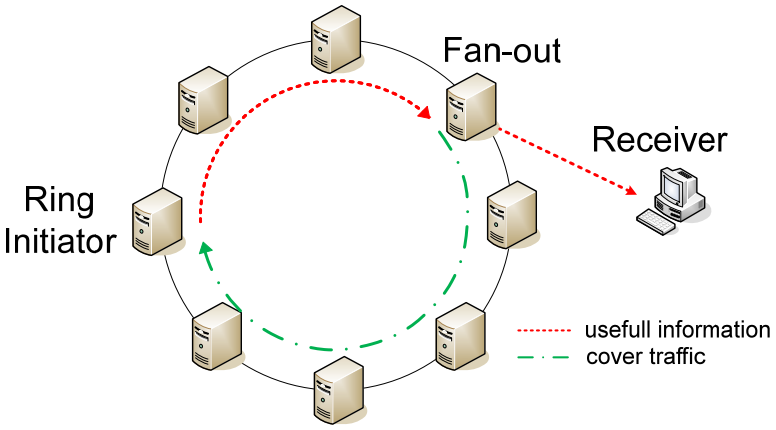

Figura 4 – Realização de comunicações anónimas ponto-a-ponto através de uma Mix Ring

A definição desta nova arquitectura foi efectuada tendo em conta requisitos de tolerância a falhas em que os Counters no anel podem falhar sendo, no entanto, necessário manter o funcionamento do anel. Relativamente à segurança do sistema considera-se que os Counters não provocam ataques de negação de serviço. Assume-se que os Counters são "cooperantes mas curiosos", querendo com isto dizer que poderão actuar em conluio e tentar capturar o máximo de informação possível mas não impedir a realização da eleição.

Esta nova arquitectura, como pode ser observado na Figura 5, mantém inalterados os dois primeiros passos da arquitectura original. No entanto, reduz-se o número de actores no sistema, uma vez que deixam de ser necessários os Anonymizers e o processo de anonimato das comunicações passa a ser desempenhado por uma espécie de Mix Ring implementada pelos servidores Counter existentes no sistema.

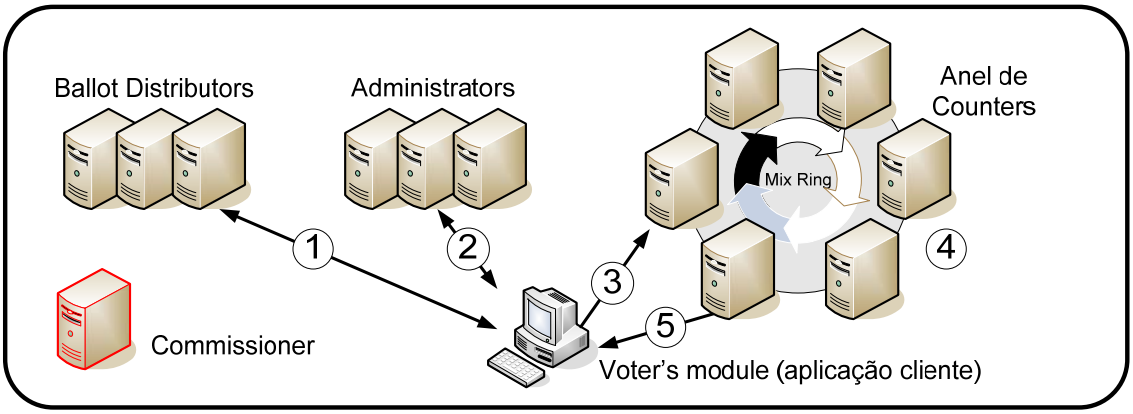

Figura 5 – Nova arquitectura para o sistema REVS

Contudo, esta nova arquitectura apresenta algumas diferenças significativas relativamente aos Mix Rings. Uma das principais diferenças é o facto do iniciador da comunicação ser externo ao anel, correspondendo ao passo três em que o votante faz a submissão da mensagem contendo o seu voto para o anel de Counters. Esta mensagem é construída por camadas de cifra sucessivas, sendo cada camada cifrada respectivamente com a chave pública do Counter escolhido pela aplicação do votante. Assim, iremos possuir dois anéis, um em que participam todos os Counters disponíveis no sistema – ***Real Mix Ring*** (**RMR**) – e outro, responsável pelo processamento das mensagens, onde apenas se encontram os Counters escolhidos pela aplicação cliente – ***Logical Mix Ring*** (**LMR**). Uma vez que o segundo anel corresponde a um subconjunto do primeiro, passamos a chamar a esta topologia **RoR** (***Ring over Ring***). Esta distinção pode ser observada na figura seguinte.

Dos Counters escolhidos para constituírem o LMR a aplicação do cliente escolhe aleatoriamente e identifica quais deverão executar os papéis principais: *Store Counter* (recepção e armazenamento do voto) e *Receipt Counter* (devolução do recibo ao votante).

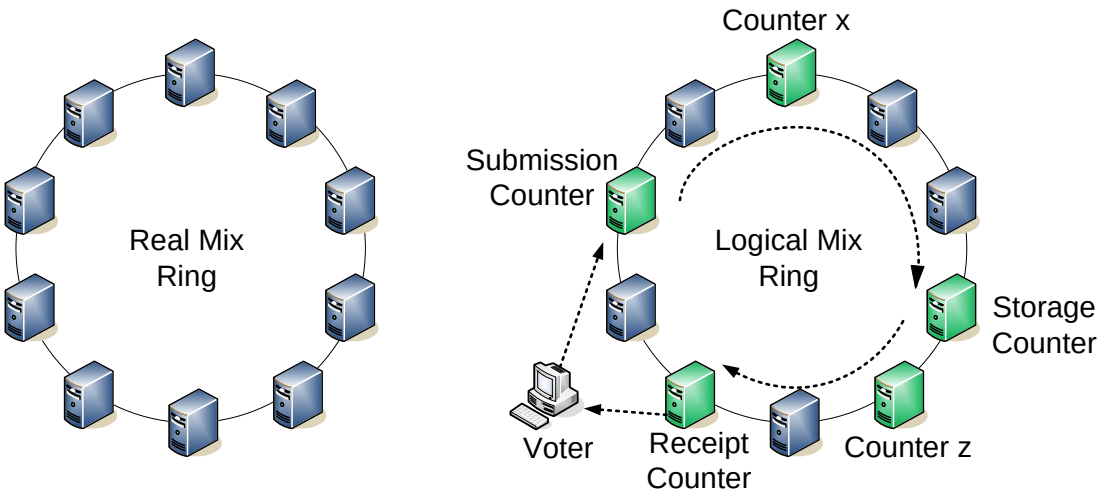

Figura 6 – Topologia RoR – distinção entre *Real Mix Ring* (RMR) e *Logical Mix Ring* (LMR)

A escolha do LMR poderá condicionar a obtenção de sucesso na execução de todo o processo. A nova arquitectura possui tolerância limitada a falhas. Limitada no sentido de que se as falhas ocorrerem em Counters pertencentes exclusivamente ao RMR, e não a um LMR, o sistema consegue recuperar e processar as mensagens com sucesso (ver Figura 7).

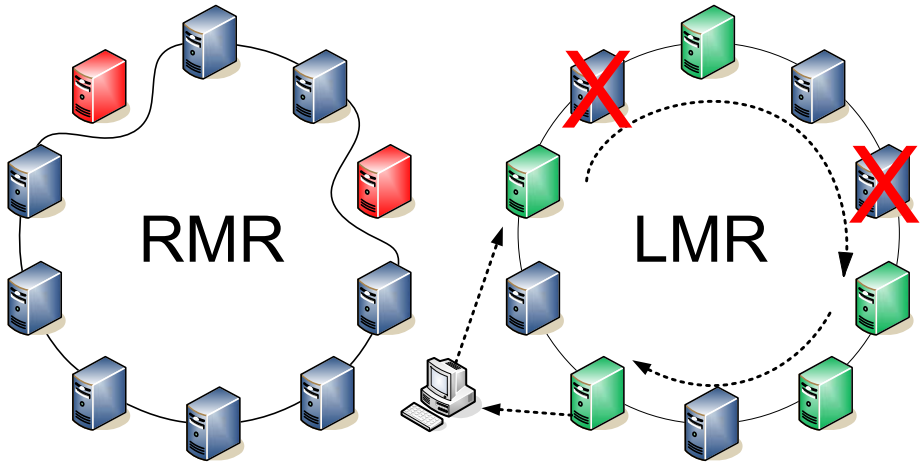

Figura 7 – Sucesso na recuperação de uma falha: O RMR recupera e mantém-se o LMR necessário aos votos em circulação

Contudo, se o Counter em que ocorreu a falha pertencer também ao LMR, conduz a uma situação em que não será possível recuperar o processamento da mensagem em virtude da mensagem ser cifrada com a sua chave pública; este caso pode ser observado na figura seguinte.

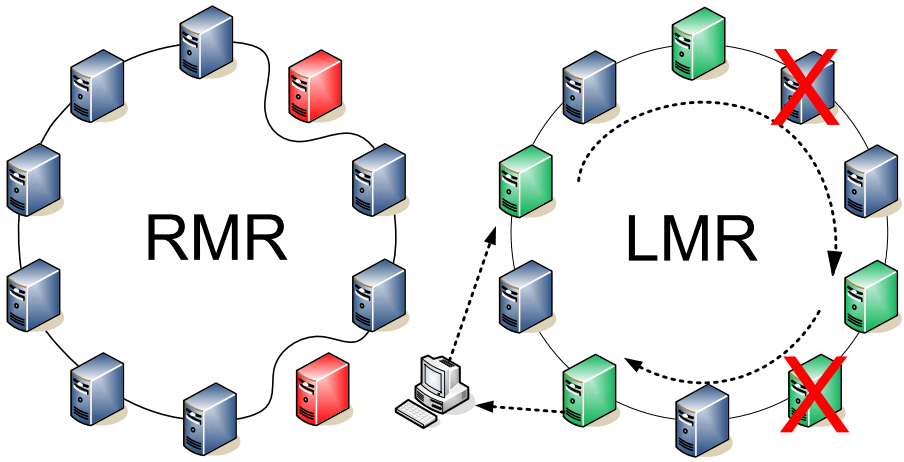

Figura 8 – Sucesso na recuperação de uma falha: O RMR recupera e mas não se mantém o LMR necessário aos votos em circulação

Cada Counter poderá realizar uma das seguintes operações sobre a mensagem recebida:

1. reenvio para o Counter seguinte;
2. processamento da mensagem (decifra e eventual execução de mecanismos de *fan-out*);
3. remoção de lixo em circulação (*garbage collection*).

A realização de operações de *garbage collection* é baseada em mecanismos similares a "*dejá vu*", isto é, cada Counter regista a observação de cada mensagem recebida. Deste modo, é possível a contagem do número de vezes que a mesma mensagem passa por um determinado Counter e remover a mesma ao fim de N observações. Uma vez que as mensagens são construídas por cifras em camada, uma mensagem só poderá ser vista como sendo a mesma se durante uma volta completa no anel não tiver sido processada, o que equivale a dizer que o Counter necessário não está disponível no RMR. Para fazer a gestão dos registos de *garbage collection*, assim que uma mensagem é removida todas as entradas no registo anteriores a essa mensagem podem ser eliminadas uma vez que correspondem a ocorrências anteriores e foram removidas por outro Counter ou eventualmente o Counter recuperou da falha a tempo de realizar o processamento da mensagem.

À semelhança dos Mix Rings, a comunicação com o destinatário é efectuada através de um processo de *fan-out*. Também aqui existem diferenças, uma vez que são necessários dois mecanismos de *fan-out* distintos:

1. o primeiro para armazenamento do voto, no passo 4 da Figura 5, o qual designamos por *fan-out* interno em virtude do destinatário pertencer ao anel;
2. o segundo, no passo 5 da mesma figura, para envio de um recibo para o votante e designado por *fan-out* externo por analogia com o primeiro.

O recibo enviado para o votante, designado por **recibo inócuo**, não compromete a sua privacidade em virtude de o mesmo não possuir qualquer correlação com o seu voto submetido. Este recibo corresponde a um vector de bits gerado aleatoriamente pela aplicação cliente e encontra-se inserido na camada interior da mensagem. A função deste recibo é a de dar a conhecer ao votante o resultado do processo de submissão, conseguido através da comparação de ambos os recibos: o gerado pela aplicação e o devolvido pelo anel. Caso ambos sejam iguais, significa que o processo de anonimato na comunicação decorreu com sucesso. Assumindo a existência de pelo menos um Counter honesto antes de cada operação de *fan-out*, o Counter escolhido para armazenamento do voto não tem acesso a qualquer informação que o possa ajudar a decidir pela aceitação ou não do voto, como tal, este recibo poderá ser considerado como uma prova, ainda que fraca, da inclusão do voto na contagem final.

Capítulo 3

Trabalho relacionado

O recurso a anéis virtuais para troca de informação não é exclusivo de nenhum tipo de comunicação em particular, surgindo nas mais diversas aplicações com o objectivo de melhorar o desempenho global dos sistemas. Assim, o estudo de trabalhos anteriores dividiu-se por áreas diversas, sendo necessário, contudo, manter sempre presentes as reais necessidades dos sistemas para que foram desenvolvidos, bem como os nossos requisitos operacionais antes descritos.

3.1. Anéis lógicos para comunicações multicast

O VRing [8] é um protocolo para ***Application Layer Multicast*** (**ALM**) baseado numa rede em anel. Uma das principais diferenças entre o *multicast* IP e o ALM é que, neste último, a informação apenas é replicada para os restantes elementos do grupo através de cada um dos membros, ou seja deixam de ser os *routers* IP ao longo da rede os responsáveis pela replicação e gestão da informação. Assim, todos os pacotes de dados são enviados a um dado subscritor do serviço com recurso a comunicações *unicast*, sendo depois este o responsável pelo reencaminhamento da informação para o membro seguinte e assim sucessivamente até atingir todos os membros do grupo.

Deste modo, são identificadas duas fases distintas na utilização do anel virtual, compreendendo a construção e a manutenção da operacionalidade do mesmo. Estas fases são descritas a seguir.

3.1.1. Iniciação/Construção do anel

Inicialmente cada membro deverá registar-se no ***Rendevouz Point*** (**RP**) como líder, sendo que este RP deverá ser conhecido por todos os subscritores deste grupo de *multicast*.

Para efectuar o registo são enviadas mensagens do tipo LEADER_REGISTER para o RP; este tipo de mensagens é também utilizado para a renovação do mesmo registo com um período T_{Leader}. Estes registos podem ser removidos tanto pelo RP, devido a ter sido atingido o prazo de validade definido como igual a $T_{Leader\_Lifetime}$ segundos, como pela recepção de uma mensagem do tipo RETIRE submetida pelo membro responsável pelo registo. Como resposta às mensagens de LEADER_REGISTER, o RP regista este membro na sua lista de líderes e procede ao envio de uma mensagem do tipo LEADER_REGISTER_REPLY, nesta comunicação o RP envia ao membro a identificação de todos os líderes registados nesse preciso momento.

Durante este processo de iniciação do anel, pode ocorrer o registo de dois tipos de componentes, cada um deles com o respectivo líder. Estes poderão corresponder a membros isolados ou conjuntos de pelo menos dois membros ligados em anel. Cada membro é responsável pelo armazenamento dos identificadores dos membros **seguinte**, **anterior** e **líder**. Deste modo, para um membro isolado as duas primeiras variáveis serão iniciadas com o valor nulo sendo o próprio identificado como líder para esse componente.

Esta fase termina apenas quando a lista existente no RP estiver reduzida a uma única entrada e durante um período de tempo predeterminado não ocorrer o registo de nenhum novo líder. Assim, torna-se necessário proceder à negociação entre os vários líderes de forma a convergir para um único anel e atingir este objectivo.

Deste modo, quando um líder **u** sabe da existência de outros líderes, ele escolhe o que estiver mais próximo de si (utilizando os critérios previamente definidos), neste caso **v**, procedendo então ao envio de uma mensagem do tipo **JOIN(Pred=u.pred, Succ=u.succ)**. Contudo, num dado instante um líder apenas pode estabelecer comunicação com um único

líder. Assim, caso o líder destino esteja ocupado, a comunicação é rejeitada e terá de ser escolhido outro líder para tentar estabelecer a comunicação pretendida.

Em resposta à mensagem de JOIN, é enviada uma mensagem do tipo **JOIN_OK(Pred=v.pred, Succ=v.succ)** ou então, no caso de **v** já não ser líder, **JOIN_DECLINE(Leader=v.leader)**.

Nas imagens seguintes pode-se facilmente visualizar os procedimentos executados para as várias situações por diversos tipos de elementos. Assim, na Figura 9 apresenta-se a situação de junção entre dois líderes isolados.

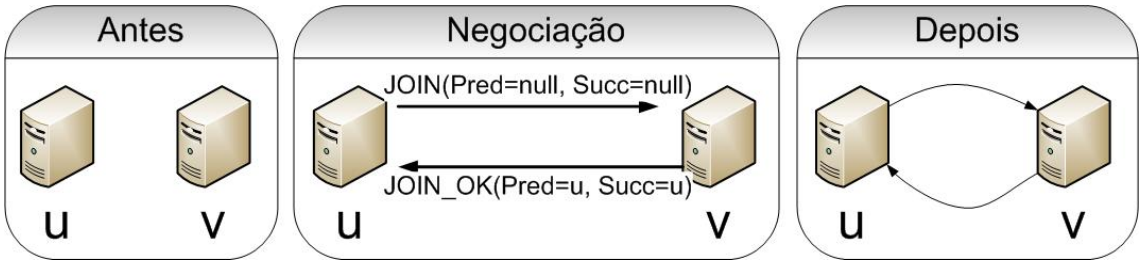

Figura 9 – Junção entre dois líderes isolados

Quando a junção é efectuada entre um líder isolado e outro líder integrado num anel, podem ocorrer duas situações: o pedido é realizado pelo líder isolado através do envio de uma mensagem JOIN(null,null) indicando que o mesmo se encontra isolado (ver Figura 10); ou no caso do pedido ser realizado pelo líder integrado no anel, a mensagem enviada corresponderia a JOIN(p,s).

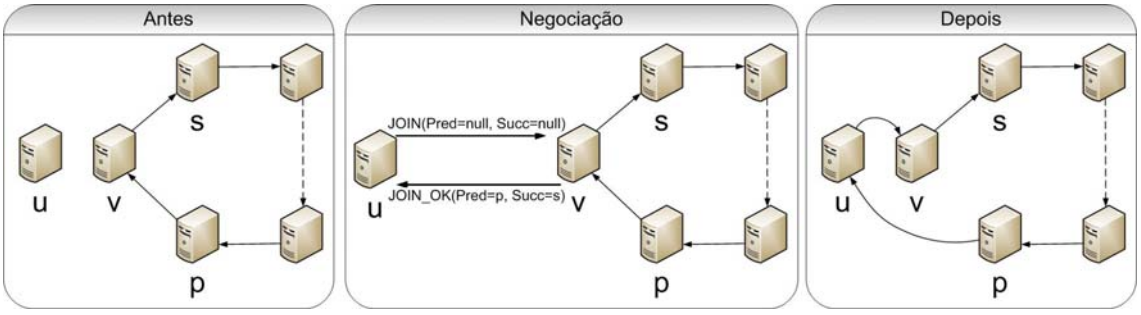

Figura 10 – Junção entre um líder isolado e outro integrado num anel

No caso da Figura 11, ambos os intervenientes são líderes de anéis disjuntos. Deste modo, após a realização do pedido de JOIN, os anéis são unificados seguindo a ordenação dos identificadores.

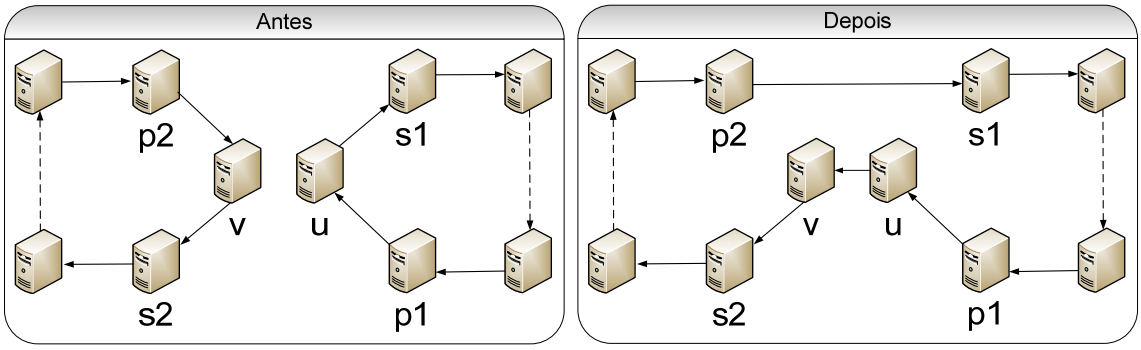

Figura 11 – Junção entre dois líderes de anéis disjuntos

3.1.2. Manutenção do Anel

O segundo momento corresponde à operação do anel, mais concretamente à sua manutenção através da monitorização do estado da rede num dado momento. Deste modo, situações de falha de um ou mais membros podem ser detectadas com recurso a um protocolo de troca de mensagens do tipo HELLO.

Este protocolo requer que cada membro envie periodicamente, a cada T_{HELLO} segundos, uma mensagem para o seu nó seguinte. Aquando da recepção dessa mensagem o membro, **u**, responde com uma mensagem de **HELLO_REPLY(Succ=u.succ)**.

Sempre que ocorre a troca das mensagens de HELLO e HELLO_REPLY o receptor verifica se a origem da mesma corresponde ao seu predecessor ou ao seu sucessor respectivamente, de forma a poder saber se o seu sucessor ainda é membro do grupo de *multicast*.

Se ao fim de MAX_NUM_NO_HELLOREPLY tentativas não tiver sido obtida qualquer resposta o nosso membro assume que o seu sucessor está desactivado ou abandonou o grupo.

Cabe ao predecessor a responsabilidade na recuperação das falhas detectadas durante a execução do protocolo de HELLO. Assim que for detectada a falha de um sucessor, o predecessor envia, para o sucessor do seu sucessor, conhecido durante a troca

de mensagens de HELLO, uma mensagem de LINK_REPAIR à qual, no caso de sucesso, obterá como resposta uma mensagem do tipo LINK_REPAIR_ACK.

Contudo, poderão surgir problemas quando ocorre uma falha em dois membros consecutivos do anel lógico. Para lidar com esta situação os autores propõem a utilização de listas de comprimento **m** com as informações relativas aos **m** membros seguintes.

3.2. Anéis lógicos para comunicações peer-to-peer

Para além da utilização de anéis virtuais nas comunicações *multicast*, existem outros estudos onde o objecto de estudo é a comunicação *peer-to-peer*. Entre eles encontramos o ***Virtual Ring Routing*** (**VRR**) [9], centrado nas redes sem fios, que implementa um protocolo de encaminhamento baseado nas *Distributed Hash Tables* (**DHT**). As DHT são utilizadas essencialmente para obter as seguintes características:

- Descentralização: os nós implementam o sistema de uma forma colectiva sem qualquer necessidade da existência de um coordenador central;
- Escalabilidade: o sistema deverá funcionar de uma maneira eficiente e correcta independentemente do seu número de nós;
- Tolerância a falhas: o sistema deve ser fiável independentemente do estado da rede, introdução, remoção ou falha de nós.

A cada nó é atribuído um identificador fixo e único, gerado aleatoriamente, que não depende da sua localização física. Este identificador é utilizado para organizar o anel virtual, correspondendo o mesmo a uma ordenação dos nós relativamente aos seus identificadores, conforme pode ser observado na Figura 12.

Para a aplicação desta técnica cada nó deverá comunicar apenas com alguns dos intervenientes na rede de modo a limitar a propagação de alterações através da rede de cada vez que ocorra uma mudança nos participantes no sistema. Assim, para garantir a integridade do anel virtual, durante a ocorrência de falhas nos nós ou nas ligações, cada nó armazena um conjunto da sua vizinhança virtual (**vset**). Este conjunto, com **r** elementos, é composto por **r/2** nós anteriores e por **r/2** nós seguintes.

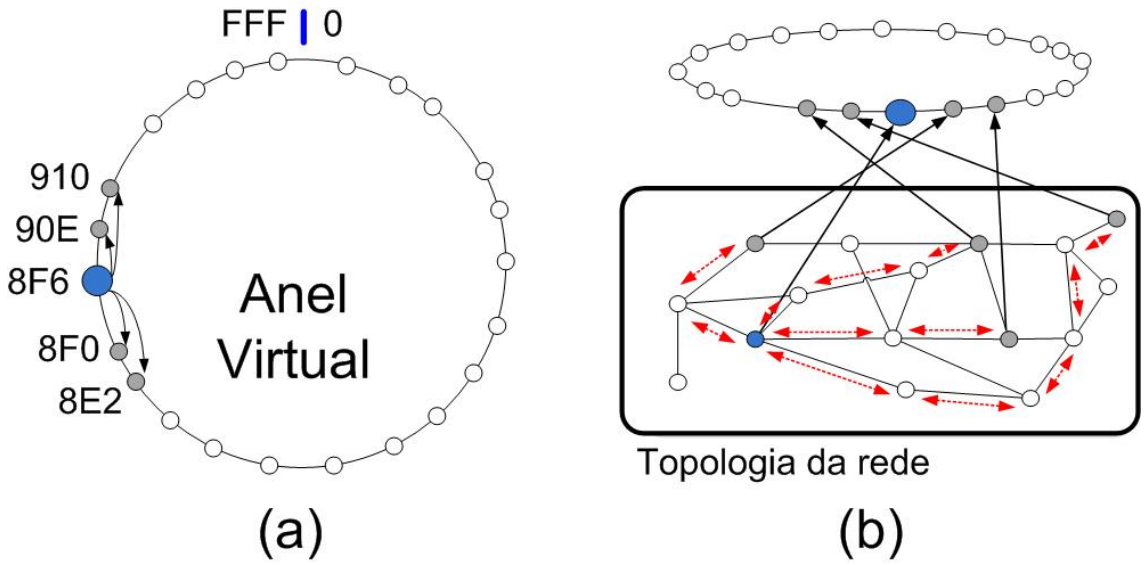

Figura 12 – Relação entre o anel virtual e a topologia física da rede

Sendo que, cada elemento pertencente ao anel possui um conhecimento limitado sobre o mesmo, conhecendo apenas os seus vizinhos a montante e jusante até uma distância igual a r/2, torna-se necessário descrever procedimentos que possibilitam a comunicação entre os elementos constituintes do anel, independentemente de pertencerem ou não ao vset de um dado elemento. Surge assim os **vset-paths**, que não são mais do que a definição de rotas para cada um dos seus nós vizinhos existentes no vset. Todas as rotas estabelecidas entre um nó e os seus vizinhos são guardadas numa tabela de encaminhamento, a qual deve ser actualizada sempre que se registarem alterações no vset. Estas rotas definidas nos vset-path são simétricas, uma vez que se um nó x pertencer à vinhança (vset) de outro nó y o contrário também vai ser verdadeiro, pelo que a rota definida para um vai ser simétrica da rota definida pelo outro.

A necessidade da existência de rotas unicamente para os vizinhos pertencentes ao vset prende-se com o facto da ordenação do anel ser baseada na ordenação dos identificadores de cada nó. Deste modo, sempre que um nó pretenda comunicar com outro não pertencente ao seu vset, este utiliza o vset-path para o seu vizinho com o identificador mais próximo do identificador do nó de destino. Do mesmo modo, sempre que um nó receba uma mensagem para outro nó que não pertença à sua vizinhança, efectua os mesmos procedimentos apresentados anteriormente através do reencaminhamento da

mensagem pelo seu vset-path mais próximo do destino final. Os nós do VRR simplesmente enviam os pacotes de dados, destinados a um determinado nó, recorrendo ao reencaminhamento dos mesmos para o seu nó seguinte, utilizando o caminho que leva ao nó com o identificador mais próximo do identificador de destino. Note-se que a comunicação no VRR é bidireccional.

As DHT podem, também, ser utilizadas para garantir a robustez contra a participação de nós maliciosos, bem como para o anonimato dos participantes no sistema de comunicação. Deste modo, é sugerida a utilização do valor de *hash* da chave pública de cada nó como o seu identificador, em vez da utilização de números inteiros gerados aleatoriamente.

Outro exemplo da utilização de uma topologia em anel para as comunicações *peer-to-peer*, o ***Self-Stabilizing Structured Ring*** (**SSSR**), foi proposto por Shaker e Reeves [10]. Para a construção do anel baseiam-se no critério de proximidade entre os vários intervenientes nessa rede, pelo que o SSSR não necessita de um coordenador central para administrar a rede.

Este protocolo apresenta robustez relativamente a falhas nos nós de comunicação (*peers*) sem ser necessária a intervenção do nó que abandona a rede. Cada *peer* interveniente na rede deve armazenar as seguintes informações relativas ao anel em que participa:

- Conjunto da vizinhança actual (**Γ**);
- Conjunto dos *peers* devolvidos pelo **Closer-Peer Search (W)**;
- Conjunto dos *peers* obtidos através do protocolo **Search Monitor (B)**;
- *Peer* escolhido aleatoriamente pelo seu sucessor actual, ou os *peers* devolvidos pelo sistema de arranque (*bootstrapping*).

Deste modo, não existe a necessidade de comunicar as alterações verificadas na rede a todos os elementos constituintes do anel de comunicação. Periodicamente cada nó irá executar uma pesquisa na rede, *Closer-Peer Search*, de forma a tentar encontrar um *peer* para seu sucessor que esteja mais próximo de si do que o seu sucessor actual (ver Figura 13). Para a realização deste procedimento, o *peer* **x** contacta o *peer* **s** escolhido

aleatoriamente pelo seu sucessor ou então obtido a partir do serviço de arranque. Assim que o *peer* **s** receber o pedido, reenvia-o para o seu vizinho que estiver mais próximo de **x**, o qual irá dar o mesmo tratamento à mensagem, aquando da sua recepção, com o objectivo de chegar o mais perto possível do *peer* **x**. Esta procura termina quando esta mensagem chegar a um *peer* **u** e este verificar que o *peer* iniciador **x** encontra-se mais próximo de si do que qualquer um dos seus *peers* vizinhos. O *peer* **u** envia então ao *peer* **x** o endereço e o identificador do seu sucessor, para que **x** possa adicionar ao seu conjunto W. Caso a rede esteja correctamente configurada, esta procura terminará no *peer* **x**, não produzindo qualquer efeito sobre a configuração da rede.

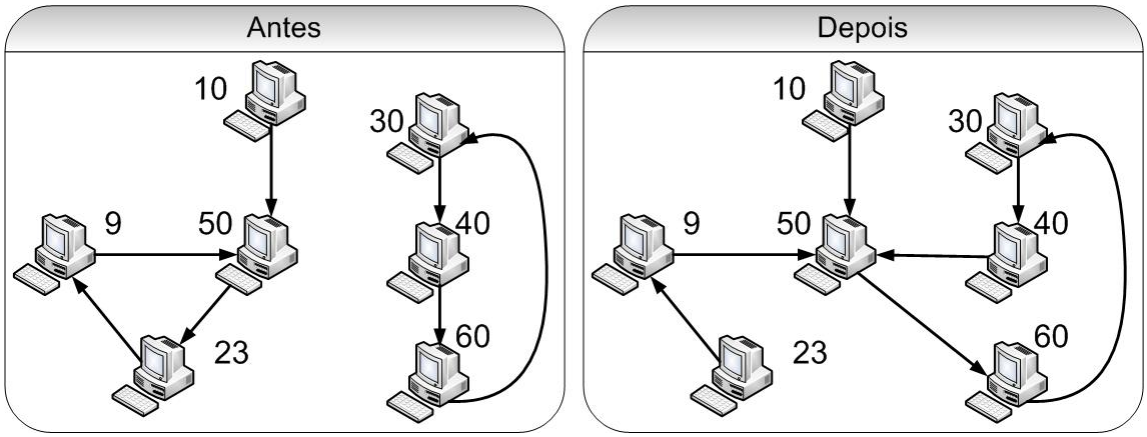

Figura 13 – Exemplo de execução do protocolo Closer-Peer Search

3.3. Comparação dos modelos estudados

Da apresentação antes feita sobre anéis de comunicação pode-se verificar que, apesar de recorrerem a técnicas diferenciadas no que respeita à criação do anel, existem algumas propriedades para as quais apresentam comportamentos similares. Estas semelhanças reflectem os requisitos e as preocupações relacionadas com a segurança nos sistemas para os quais foram desenhados. Um breve resumo da comparação entre os sistemas pode ser observado na Tabela 1.

Assim, para todos os sistemas apresentados não existem preocupações relacionadas com a dupla inserção de nós, uma vez que nas aplicações para que foram desenhados esta ocorrência não representaria um problema. No entanto, para o sistema VRR é proposta a utilização, para o identificador do nó, de um *hash* da sua chave pública caso se pretenda

autenticar os nós. Contudo, não estão previstas limitações de acesso aos sistemas, a participação está aberta a todos os nós que desejem fazer parte do mesmo.

Devido ao conceito das comunicações *multicast* e *peer-to-peer*, não se encontra definido qualquer limite relativamente ao número máximo de nós que o sistema possa suportar. Do mesmo modo, para estes tipos de comunicações não existem preocupações com a segurança das comunicações realizadas, pelo que não existe a necessidade da implementação de canais de comunicação seguros entre os nós constituintes dos sistemas.

Tabela 1 - Comparação dos modelos estudados

Modelo	*VRing [8]*	*VRR [9]*	*SSSR [10]*
Aplicação	*Multicast*	*Peer-to-peer*	*Peer-to-peer*
Inicialização	Registo RP	Atribuição dos identificadores	*Bootstrapping* e Closer-Peer Search
Ordenação	Critérios de proximidade	Identificador	Identificador de proximidade física
Coordenação Central	Sim (líder)	Não	Não
Manutenção	Protocolo HELLO	Recurso às DHT	Closer-Peer Search
Tolerância a falhas	Sim	Sim	Sim
Detecção de falhas	Protocolo HELLO	Protocolo HELLO	N/D
Ponto Singular de Falha	Sim	Não	Não
Dupla inserção de nós	Sim	Sim	Sim
Inserção de nós não autorizados	N/A	N/A	N/A
Utilização de canais seguros	Não	Não	Não
Número total de nós	Indeterminado	Indeterminado	Indeterminado

Para a criação dos anéis lógicos, todos os sistemas recorrem a sistemas auxiliares. No caso do sistema VRing é necessário proceder ao registo perante o *Rendevouz Point*, responsável pela identificação dos líderes. Já o VRR requer um mecanismo de atribuição de identificadores únicos aos participantes. No caso do SSSR necessita-se, para além dos identificadores baseados num critério de proximidade, de um sistema de arranque que auxilie os nós durante o processo inicial de adesão ao sistema. Mais uma vez, existe um ponto comum a todos os sistemas, a ordenação dos nós no anel segue critérios predefinidos

que representam a proximidade entre eles ou apenas uma simples ordenação dos identificadores atribuídos a cada nó.

Para a detecção da ocorrência de falhas é utilizado um protocolo do tipo HELLO tanto para o VRing como para o VRR. No caso do SSSR apenas é referido que o mesmo é tolerante a falhas sem que seja mencionado o mecanismo para a detecção e correcção das mesmas. Para a manutenção do estado de operação do anel (remoção e/ou inserção de nós) são utilizados os mesmos mecanismos de detecção de falhas em conjunto com os procedimentos descritos aquando da iniciação do anel.

Contudo, considera-se que o VRing possui um ponto singular de falha, o *Rendevouz Point*, apesar de serem implementadas medidas para a substituição do membro de coordenação central, o líder, no caso de ocorrência de uma falha no mesmo.

Capítulo 4

Requisitos operacionais

4.1. Tolerância a Falhas

Um dos requisitos operacionais colocados ao anel de Counters é tolerância a falhas. Nomeadamente, o anel deverá cumprir a sua missão caso ocorra um conjunto limitado de falhas nos Counters. Nesse sentido, o anel deverá estar sempre em funcionamento com todos os Counters disponíveis. Caso um Counter falhe, o anel deverá detectar a sua ausência e os Counters adjacentes ao que falhou deverão ligar-se entre si para colmatar a falha. Após a recuperação de um Counter o sistema deverá possibilitar a sua inserção no anel existente, respeitando a ordenação em vigor.

Os votos gerados pelos votantes são preparados para ser processados por um subconjunto de Counters do anel, designado por *Logical Mix Ring* (**LMR**). Este processamento consiste na sua decifra e eventual interpretação caso tenha de ocorrer um *fan-out* ou eliminação. Os demais Counters pelos quais o voto passará, durante o seu percurso ao longo do anel, limitam-se a reenviá-lo para o Counter seguinte sem o processar. Caso um Counter do LMR de um voto falhe, o voto não pode ser processado e fica a circular no anel até que uma das seguintes situações ocorra:

1. O voto é removido por um qualquer Counter após ter circulado no anel, um número predefinido de vezes, sem ser alterado. Esta política de remoção de lixo (*garbage collection*) evita a circulação infinita de mensagens não processáveis no anel devido à falha de um Counter. O votante poderá recuperar desta situação reenviando um novo voto mas usando um LMR diferente para o processar.
2. O Counter recupera, insere-se no anel e processa o voto. Se entretanto o votante desistiu de esperar e enviou um novo voto com um novo LMR pode acontecer que sejam armazenados dois votos, eventualmente em Storage Counters diferentes. Porém, tal não é um problema, antes pelo contrário, pode-se ganhar redundância no armazenamento, porque o REVS permite detectá-los aquando da contagem final. Este reconhecimento é realizado através de uma marca identificadora aleatória por voto (*bit commitement*) [6]; cada votante apenas consegue gerar um voto válido, o qual tem de possuir essa marca, logo votos iguais com a mesma marca são réplicas do voto do mesmo votante.

Os Counters estarão naturalmente associados a interesses em confronto (por exemplo, forças partidárias), de forma a garantir uma repartição de poder sobre o processo de submissão anónima de votos e garantir a sua correcção. Note-se que o voto de um votante pode ser associado à sua máquina se existir um conluio entre todos os Counters do LMR usado e o Submission Counter contactado. Portanto, por cada eleição existe um número finito de Counters, repartidos por vários interesses em confronto e eventualmente entidades independentes e supostamente idóneas, devendo cada Counter possuir uma identidade, que basicamente consiste num par de chaves assimétricas; esse par de chaves permitir-lhe-á identificar-se perante os seus pares e processar os votos submetidos pelos votantes.

4.2. Ataques e comportamentos maliciosos

Krutz e Vines [11] propuseram uma classificação para os tipos de ataques a que uma rede de computadores está sujeita. Esses ataques foram assim agrupados em categorias não muito rígidas em virtude das constantes evoluções que ocorrem nos dias de hoje. As categorias identificadas são as seguintes:

- Categoria A – Acesso não autorizado a serviços de rede restritos devido ao contorno das políticas de segurança.
- Categoria B – Utilização não autorizada da rede para outros fins que aqueles para que ela foi concebida e disponibilizada.
- Categoria C – Escuta não autorizada de comunicações (*eavesdropping*):
    - *Eavesdropping* passivo – monitoriza e lê as comunicações para as quais não possui autorização por parte do emissor e do receptor;
    - *Eavesdropping activo* – nesta situação o atacante realiza as operações identificadas acima mas gera tráfego para aumentar a sua probabilidade de sucesso.
- Categoria D – Negação de prestação de serviço (DoS).
- Categoria E – Intrusão na rede.
- Categoria F – Detecção de atributos ou presença (*probing*).

No modelo de segurança considerado neste trabalho assume-se que os Counters são "cooperantes mas curiosos". Tal significa que se assume que os Counters não provocam deliberadamente ataques de negação de prestação de serviço (categoria D) – por exemplo, alterando aleatoriamente todas as mensagens que por si passam – mas que podem tentar explorar todas as facilidades dadas para, sozinhos ou em conluio, obter o máximo de informação que lhes permita quebrar o anonimato na submissão dos votos. Deste modo, iremos estar interessados em controlar os ataques inseridos nas categorias A e C, sendo nesta última, para o nosso caso, apenas verificado o eavesdropping passivo. Como vimos acima a alteração das mensagens pode ser considerada como um ataque DoS.

A comunicação entre Counters precisa de ser confidencial, caso contrário um observador pode facilmente concluir quais os votos que cada Counter processou, bastando para tal comparar o tráfego de entrada com o de saída. A confidencialidade deverá ser fornecida em cada comunicação *peer-to-peer*, e não globalmente, para evitar que qualquer um dos Counters se constitua em observador do tráfego entre outros Counters. Mais ainda, as chaves de sessão usadas na comunicação entre Counters deverão possuir segurança

futura perfeita (*perfect forward secrecy*). Ou seja, o seu secretismo não deverá depender da confidencialidade de segredos de longa duração, tais como pares de chaves assimétricas dos Counters. Caso contrário, o tráfego entre certos Counters poderia ser guardado e mais tarde decifrado e analisado aquando da descoberta de segredos de longa duração relacionados com a geração da respectiva chave de sessão.

O anel deverá ser apenas formado por Counters pertencentes às entidades autorizadas para o efeito, o que significa que na formação do anel deverá ocorrer uma autenticação de interlocutores de forma a evitar a intromissão de estranhos no mesmo. Para além disso, cada entidade autorizada só poderá inserir no anel uma instância de Counter. Caso fosse possível inserir mais do que uma instância, qualquer entidade autorizada poderia inundar o anel com Counters para monitorizar outros Counters, colocando-se imediatamente antes e depois dos mesmos.

A posição relativa dos Counters no anel não deverá ser arbitrária. Caso contrário, dois Counters actuando em conluio poderiam facilmente rodear um outro Counter para melhor interpretar a sua actividade de processamento de mensagens, identificando o possível armazenamento de um voto ou o envio do recibo através da comparação das mensagens à entrada e saída do Counter alvo. Para evitar este problema os Counters deverão ter uma posição relativa fixa no anel e essa posição deverá mudar periodicamente para evitar que o cenário de ataque em conluio acima referido possa ocorrer durante toda a eleição.

Finalmente, não deverá ser possível que dois Counters excluam deliberadamente outros Counters do anel situados entre si. Essa exclusão poderia ser usada para obter o cenário de ataque em conluio acima descrito, mas pode igualmente conduzir a cenários de negação de prestação de serviço caso sejam excluídos Counters usados nos LMR dos votos submetidos. Como anteriormente assumimos que o modelo de segurança excluía ataques à prestação de serviço pelos Counters, assume-se consequentemente que os Counters não excluem deliberadamente outros Counters do anel.

Capítulo 5

Real Mix Ring (RMR)

Ao longo deste capítulo será apresentada a modelação e implementação da arquitectura de suporte para submissão anónima de votos, sendo baseada nos Mix Rings e constituída por todos os servidores Counter autorizados para a respectiva eleição. A definição do RMR segue os requisitos impostos, no Capítulo 4 , aos servidores Counter pertencentes ao anel de comunicação.

Com base nos trabalhos analisados anteriormente procedeu-se à definição da arquitectura, respeitando as devidas diferenças relacionadas com o objectivo de cada sistema, identificando as principais etapas durante a vida de um anel de comunicação.

5.1. Construção do RMR

Comum a todos os sistemas, a construção da rede de comunicação representa a primeira etapa nas operações executadas pelo anel. Para realizar esta operação, torna-se necessário definir critérios de ordenação e a correspondente troca de mensagens iniciais.

5.1.1. Ordem relativa entre os Counters

Nos protocolos anteriormente apresentados verificou-se que não existe nenhuma restrição especial relativamente à posição dos elementos no anel, podendo a mesma ser decidida pelo próprio nó ou dada naturalmente através da ordenação de números aleatórios. Não existindo qualquer verificação relativamente à identidade dos participantes, possibilita a inclusão de réplicas na constituição do anel. Contudo, de acordo com os requisitos apresentados no Capítulo 4 este facto é crítico para o anel de Counters. Ao permitir a um Counter a utilização de réplicas ou a escolha da sua posição no anel permite-se que Counters maliciosos possam conspirar e monitorizar o processamento de mensagens por parte de um Counter honesto.

Deste modo, considera-se fundamental ser da responsabilidade de uma entidade coordenadora do processo eleitoral a atribuição da ordem relativa dos Counters no anel, impedindo os próprios Counters de participar nessa tomada de decisão.

Com esta regra não se evita a possibilidade da ocorrência das situações referidas acima, contudo, essa ocorrência deixa de depender dos Counters maliciosos e fica associada a um factor aleatório. Ou seja, continuará a ser possível que dois Counters maliciosos possam cercar um Counter honesto e monitorizar as suas comunicações durante toda a eleição. Assim, de forma a reduzir a probabilidade de ocorrência dessa situação, esta ordenação aleatória deverá variar ao longo do tempo. Se, por um lado, se aumenta a probabilidade de ocorrência desse ataque, por outro garante-se que o mesmo terá uma probabilidade baixa de ocorrer durante toda a eleição.

Deste modo, antes do início da eleição são definidos e identificados os intervalos temporais em que a eleição irá funcionar. A identificação utilizada não é mais do que a numeração dos intervalos definidos pela sua ordenação temporal. Esta informação deverá ser disponibilizada numa tabela, à qual foi dado o nome de **Timezone**, para posterior consulta por parte dos Counters. Assim, o conhecimento sobre a identificação do período temporal em vigor poderá ser obtido a com base na consulta da tabela Timezone.

Para o armazenamento desta informação o protocolo proposto recorre a tabelas em XML. Esta tabela é fornecida a todos os Counters no início da eleição, sendo as consultas realizadas sobre uma cópia local. O código apresentado abaixo exemplifica a estrutura da tabela XML para a definição dos intervalos temporais, onde cada intervalo temporal é

definido pelos campos *id, horaIni e horaFim.* O primeiro campo representa o identificador do intervalo, enquanto que *horaIni* e *horaFim* servem como delimitadores temporais do intervalo, correspondendo respectivamente à hora de início e de término da configuração.

```
<?xml version="1.0"?>
<anel>
 <intervalo>
       <id>1</id>
       <horaIni>09:00</horaIni>
       <horaFim>10:00</horaFim>
 </intervalo>
 (...)
 <intervalo>
       <id>n</id>
       <horaIni>18:00</horaIni>
       <horaFim>19:00</horaFim>
 </intervalo>
</anel>
```

Uma vez que cada intervalo encontra-se delimitado temporalmente, existe a necessidade de sincronismo dos relógios internos dos Counters participantes.

Como foi referido, a definição dos intervalos temporais existe para que a ordenação dos Counters no anel, embora seja fixa para um dado intervalo, varie ao longo do tempo. Deste modo, com a necessidade de relacionar os intervalos temporais com as sucessivas ordenações do anel surge a construção de uma nova tabela designada por **Ord_Tab**. Esta tabela possui para cada intervalo temporal, previamente definido, a respectiva ordenação do anel gerada aleatoriamente pela entidade responsável pela eleição.

Para obter a posição no anel torna-se necessário que cada Counter possua um conhecimento prévio do seu identificador. Seguindo a recomendação do sistema VRR [9], propõe-se a utilização de um método de identificação com recurso ao *hash* das chaves públicas dos intervenientes, possibilitando ao mesmo tempo o controlo do acesso ao sistema restringindo o acesso aos Counters autorizados a participar na eleição.

Podem ocorrer dois tipos de consultas sobre esta tabela: (i) consultas tendo como critério de pesquisa um identificador, as quais servem para o Counter obter a sua posição

no anel, e (ii) consultas em que o critério é a ordem relativa no anel, servindo para obter os identificadores dos Counters vizinhos de um dado Counter.

Esta operação é realizada por cada Counter sempre que se verifique o início e/ou reinício do anel ou, no caso de ter ocorrido uma falha, assim que este recupere. Dado que a posição obtida é referente a um determinado intervalo temporal, existe uma probabilidade, relativamente grande (dependendo do número de nós autorizados a participar no anel), de que a sua vizinhança varie ao longo do tempo.

No fim de cada intervalo o anel é simplesmente desfeito e refeito novamente, segundo a nova ordenação, com os Counters activos nesse instante. A Ord_Tab deverá conter as informações de ordenação relativas a todos os Counters, a partir da qual será possível obter as rotas completas (distribuição dos Counters pelo anel) de maneira a permitir para a cada Counter a compreensão da sua posição relativa em todo o anel.

A existência de uma cópia da configuração completa do anel por Counter deve-se ao facto da necessidade de impor uma ordenação fixa para cada período temporal. Deste modo, qualquer Counter poderá rejeitar um pedido de junção ao anel através da simples consulta da Ord_tab.

O formato utilizado para a construção da Ord_tab encontra-se representado no código XML apresentado a seguir. Este possibilita a pesquisa da ordem relativa de um dado Counter, bem como dos identificadores dos seus vizinhos, bastando para isso conhecer o intervalo temporal em que o anel se encontra. Assim, a ordenação para um determinado intervalo é apresentada através das sucessivas identificações dos Counters, sendo cada campo “no” caracterizado pelo identificador do Counter e a respectiva posição na ordenação do anel.

```
<?xml version="1.0"?>
<anel>
 <intervalo>
       <no>
             <id>CounterX</id>
             <ordem>1</ordem>
       </no>
(...)
       <no>
```

```
            <id>CounterZ</id>
            <ordem>n</ordem>
      </no>
</intervalo>
(...)
<intervalo>
      <no>
            <id>CounterZ</id>
            <ordem>1</ordem>
      </no>
(...)
      <no>
            <id>CounterX</id>
            <ordem>n</ordem>
      </no>
</intervalo>
</anel>
```

Contudo, antes do Counter proceder ao envio do pedido de junção ao anel, este necessita de obter a localização do Counter que ocupa a posição no anel imediatamente anterior à sua. Neste caso, as informações necessárias serão o seu endereço TCP/IP (endereço IP e porto de transporte TCP). Estes dados são obtidos através da consulta da tabela **Id_Tab**, a partir da qual poderá ser efectuada a correspondência entre o identificador do Counter e a sua localização na rede.

Nesta tabela serão apresentadas as informações relativas aos Counters autorizados a participar na eleição e necessárias para a comunicação entre eles, servindo igualmente como uma forma de restrição do acesso ao sistema por entidades não autorizadas. Um exemplo do código XML constituinte da tabela Id_Tab pode ser visualizado a seguir. O campo “counter” é definido pelos seguintes subcampos: “identificador” – representando o identificador do Counter; “ip” – correspondendo ao endereço IP do Counter; port – identificando o porto de escuta do Counter.

```
<?xml version="1.0"?>
<eleicao>
 <counter>
      <identificador>CounterX</identificador>
```

```
        <ip>192.168.1.200</ip>
        <port>4001</port>
    </counter>
    (...)
    </eleicao>
```

A Figura 14 esquematiza um resumo dos procedimentos iniciais executados durante a fase de construção do anel. Aquando do arranque do Counter x, a primeira operação que este irá executar será a determinação, através de uma consulta à tabela Timezone, do intervalo temporal em que o anel se encontra nesse preciso instante (passo 1). Após a determinação do intervalo temporal actual, utiliza essa informação para efectuar a consulta na tabela Ord_Tab (passo 2) e obter a sua posição (designada por i) a partir da qual obtém os identificadores dos Counters vizinhos (analogamente nas posições i-1 e i+1) – passo 3. Uma vez que a inserção é realizada à frente, através da troca de mensagens com o Counter anterior em termos de ordenação, o próximo passo corresponde à obtenção dos dados necessários para o estabelecimento da comunicação – obtenção do endereço IP e porto TCP (passo 4). Finalmente pode ser enviada a mensagem de junção ao anel para o Counter identificado anteriormente (passo 5).

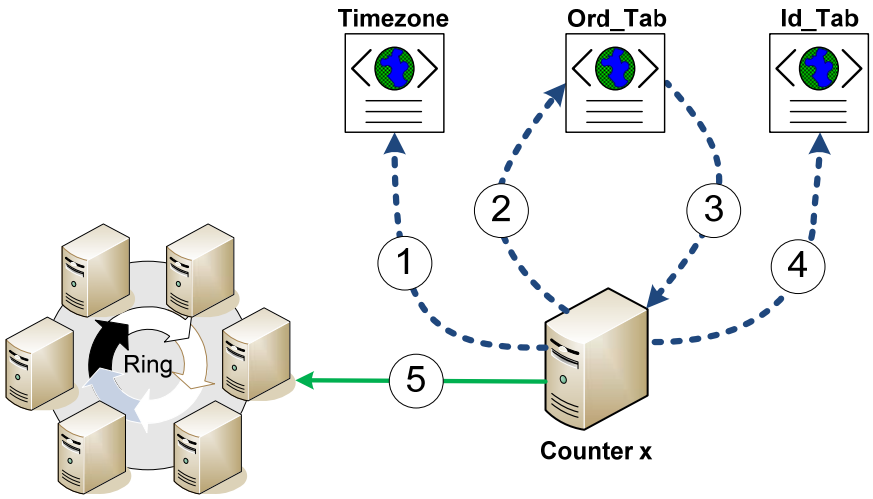

Figura 14 – Procedimentos iniciais para a criação do anel

5.1.2. Inserção de um Counter no RMR

Para facilitar a verificação e validação dos pedidos de inserção no anel, cada Counter guarda um registo dos identificadores dos Counters atribuídos pela configuração como seus vizinhos para esse período. Esta informação é utilizada para determinar o modo como o Counter responde aos pedidos recebidos ou para forçar a pesquisa pelo seu vizinho ideal.

Como já foi referido anteriormente, considera-se que a operação de inserção de um novo membro no anel é sempre realizada à frente, ou seja, o Counter que pretenda ser integrado no anel tem de contactar o seu antecessor. Assim, o Counter **x**, na posição **i**, irá contactar o seu Counter anterior, na posição **i-1**, enviando-lhe uma mensagem de JOIN_REQUEST de forma a obter a permissão de acesso ao anel. Caso este nó não lhe responda, o Counter deverá tentar comunicar com o Counter na posição **i-2** e assim sucessivamente, até obter uma resposta. Se o Counter tentou contactar todas as posições até alcançar a sua, significa que está isolado ou então não existem outros Counters activos nesse momento, pelo que o Counter deverá voltar a tentar novamente a sua inserção no anel.

O Counter anterior, assim que receber o pedido de JOIN_REQUEST verifica se o Counter **x** se encontra mais próximo de si (tendo em conta a ordenação estabelecida para o intervalo temporal actual) do que o seu nó seguinte actual. Se tal acontecer, o pedido é processado e são actualizados os dados relativos ao estado do anel. Esta operação consiste na colocação do identificador do Counter **x** como seu nó seguinte, passando assim a reencaminhar todas as mensagens que lhe cheguem por este Counter.

Como resultado de uma mensagem de JOIN_REQUEST o Counter **x** poderá (i) não receber qualquer resposta, (ii) receber uma mensagem de JOIN_OK com a indicação do identificador do seu nó sucessor actual, ou (iii) receber uma mensagem de JOIN_DECLINE a qual obriga o Counter a reiniciar a sua procura pelo Counter anterior. A Figura 15 apresenta o diagrama de fluxos para o envio das mensagens de JOIN_REQUEST do Counter **x**, na posição **i**, de forma a obter a identidade do Counter anterior actual (C_{ant}) e a identidade do Counter seguinte actual (C_{seg}).

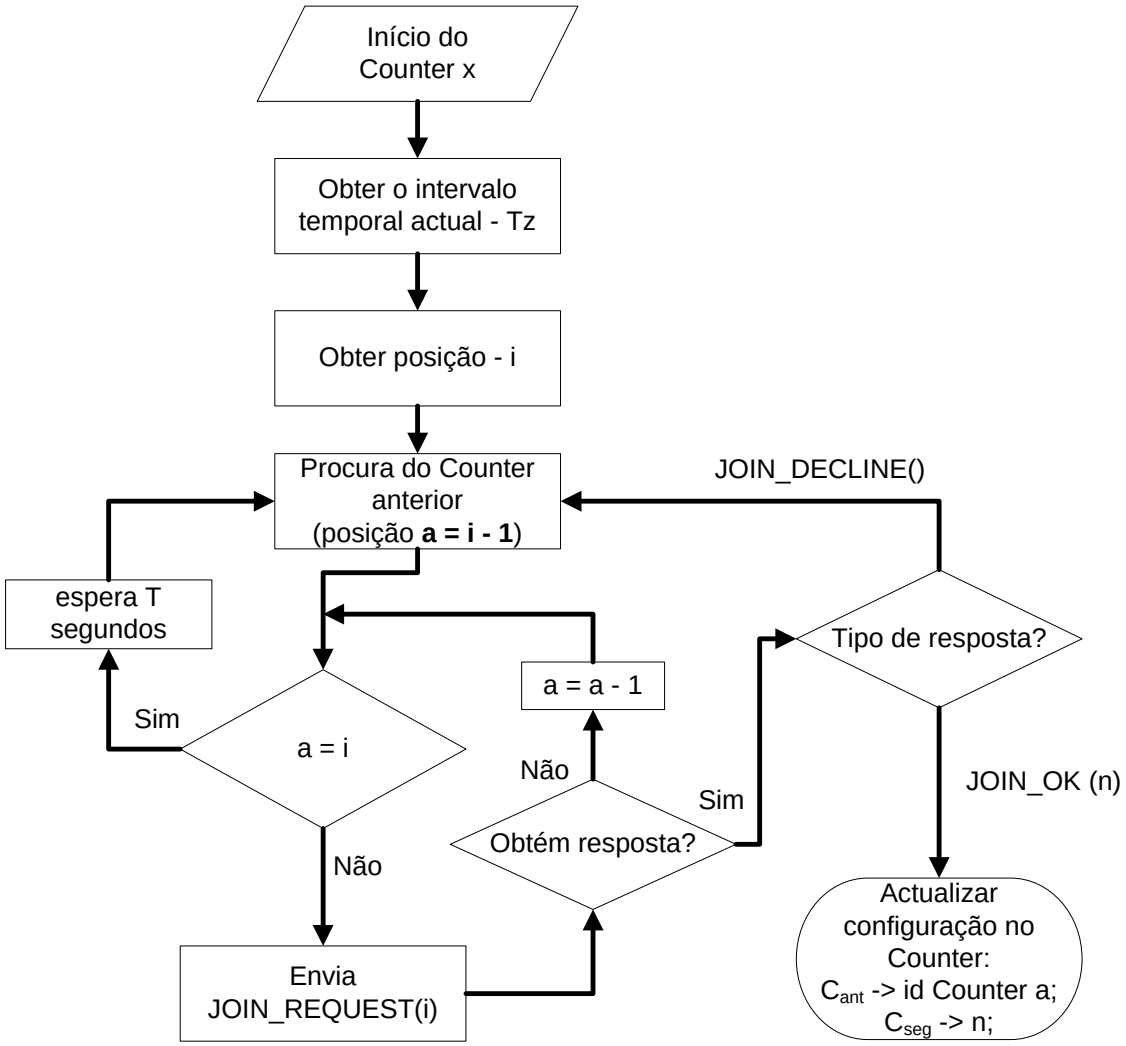

Figura 15 – Diagrama de fluxos para o envio de uma mensagem do tipo JOIN_REQUEST

Durante este processo o Counter deverá estar atento à possível ocorrência da construção de dois, ou mais, anéis distintos devido ao arranque simultâneo de dois ou mais Counters. Para lidar com esta situação, no caso em que o Counter que recebe um pedido JOIN_REQUEST verifique que o seu C_{seg} corresponde ao esperado para esse período temporal, ou está mais próximo topologicamente do esperado, responde com uma mensagem do tipo JOIN_DECLINE, obrigando o Counter que enviou o pedido a reiniciar o seu protocolo de inserção no anel. Ao mesmo tempo, no caso de verificar que o seu Counter anterior não corresponde ao definido na configuração do anel, executa regularmente os mecanismos de optimização e que o permitem aproximar o mais possível da configuração óptima (com o máximo possível de Counters activos).

Na Figura 16 apresenta-se o diagrama de fluxo das operações realizadas por um Counter, já inserido no anel, após ter recebido uma mensagem JOIN_REQUEST do Counter com identidade **x**. Caso a inserção seja aceite, com um envio da mensagem JOIN_OK, o Counter aceitante indica a identidade do Counter seguinte que mantinha até então (**C_2**), e que passará a ser o seguinte do Counter **x**.

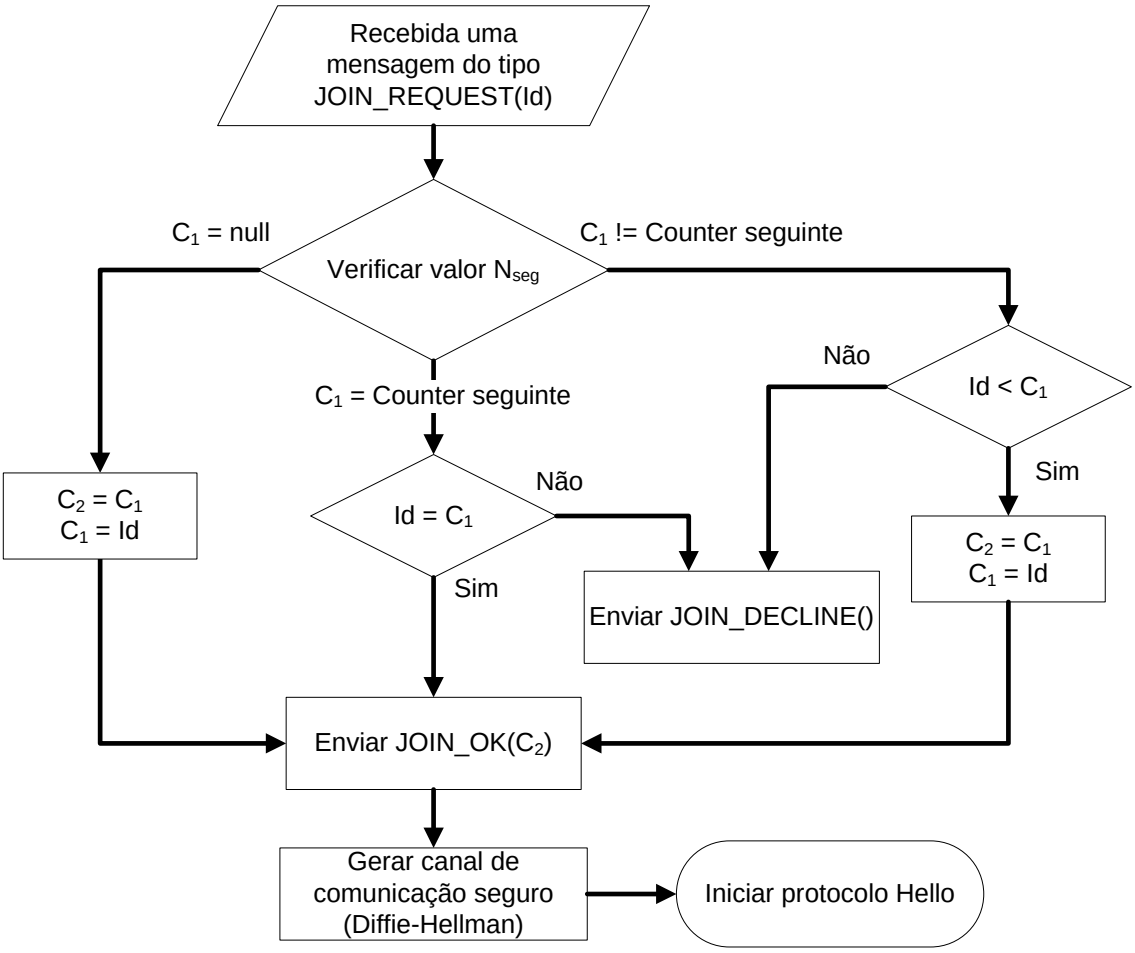

Figura 16 - Diagrama de fluxos de resposta a uma mensagem JOIN_REQUEST

5.1.3. Autenticação mútua e negociação de chaves de sessão

O último passo na construção do anel refere-se à autenticação mútua entre Counters contíguos e ao estabelecimento de um canal de comunicação seguro entre si. A autenticação impede que tenham acesso ao anel entidades não autorizadas, bem como garante que um dado Counter demonstre que realmente é quem diz ser. Já a chave de sessão permite dotar o tráfego entre Counters de confidencialidade, de acordo com os

requisitos enunciados no Capítulo 4 , impedindo a percepção das comunicações por parte de entidades externas.

As chaves de sessão devem ter segurança futura perfeita, de forma a impedir que no futuro possam ser descobertas e revelar o tráfego protegido. O método normal de obter tais chaves é através do algoritmo de Diffie-Hellman [12] e usando valor secretos efémeros. No entanto, este algoritmo não contempla a autenticação dos intervenientes. Sendo a autenticação um dos requisitos de segurança do sistema, necessita-se de uma versão alterada do algoritmo em que os valores públicos de Diffie-Hellman são trocados juntamente com uma assinatura feita pelo Counter emissor, usando para o efeito a sua chave privada. Assim, pretende-se anular a possibilidade da ocorrência de ataques do tipo "*man-in-the-middle*". Este tipo de ataque ocorre quando as trocas de mensagens não são autenticadas e existe um intermediário que se faz passar por ambos os interlocutores, podendo desse modo escutar as comunicações em ambos os sentidos.

Para a implementação deste protocolo, na geração do canal, são utilizados dois parâmetros p e g, os quais são públicos. O parâmetro p corresponde a um número primo, enquanto que o g corresponde a um número inteiro menor que p e que verifique a seguinte condição.

$$\forall n \in [1, p-1] : n = g^k \bmod p$$

Deste modo, para que dois Counters A e B comuniquem entre si através de um canal seguro, deve-se em primeiro lugar criar uma chave secreta partilhada com recurso ao protocolo Diffie-Hellman, através da execução dos passos descrito a seguir e representados na Figura 17.

Inicialmente, cada um dos Counters é responsável por gerar aleatoriamente um valor secreto. Estes valores são ambos números inteiros, considerando-se nesta representação como a e b respectivamente para o Counter A e B. Seguidamente, a partir dos valores secretos, cada Counter terá de calcular o correspondente valor público com recurso à condição apresentada acima. Assim, o valor público para o Counter A será $g^a \bmod p$, do mesmo modo o valor público para o Counter B será $g^b \bmod p$.

Após a obtenção dos valores públicos, ambos os Counters terão de proceder à autenticação dos mesmos. Uma vez que todos os Counter autorizados a participar na

eleição possuem um par de chaves assimétricas, esta autenticação dos valores públicos é realizada recorrendo à cifra com a respectiva chave privada. Posteriormente são trocados entre ambos os intervenientes os valores autenticados.

Finalmente, o Counter A processa o valor proveniente de B $g^{ab} = \left(g^{b} \bmod p\right)^{a} \bmod p$, ao mesmo tempo que o Counter B realiza a mesma operação com o valor recebido do Counter A $g^{ba} = \left(g^{a} \bmod p\right)^{b} \bmod p$. A partir deste momento considera-se que existe a partilha de uma chave simétrica e consequentemente o estabelecimento do canal anónimo para troca de informações entre ambos os Counters. Através da equação apresentada a seguir, pode-se comprovar a equivalência das operações, que permitem a obtenção da chave partilhada, executadas por ambos os Couters.

$$k = g^{ab} = \left(g^{b} \bmod p\right)^{a} \bmod p = \left(g^{a} \bmod p\right)^{b} \bmod p = g^{ba}.$$

Na figura apresentada a seguir podemos observar um esquema representativo da troca de mensagens necessária para a obtenção de uma chave secreta partilhada com recurso ao do protocolo modificado baseado no Diffie-Hellman com suporte a autenticação.

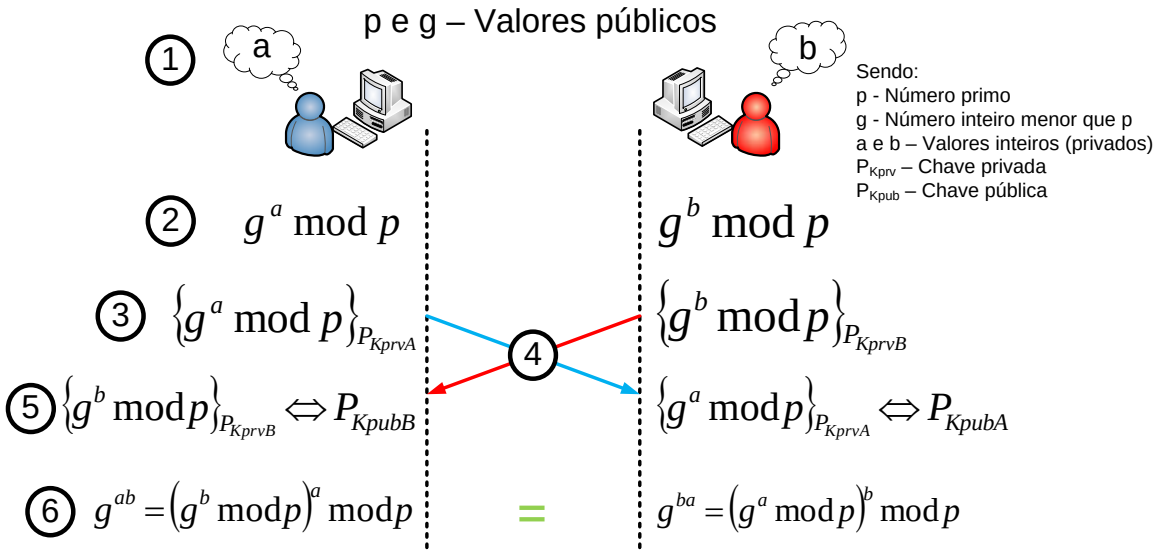

Figura 17 – Esquema de execução do protocolo Diffie-Hellman com autenticação dos valores públicos. (1) geração aleatória dos valores privados; (2) cálculo dos valores públicos; (3) cifra dos valores públicos com a chave privada; (4) troca entre os intervenientes dos valores públicos cifrados; (5) decifra dos valores públicos; (6) obtenção da chave secreta partilhada.

Com este mecanismo, a identidade do interlocutor fica associada ao canal de comunicação e não à informação veiculada através do mesmo. Ou seja, se um Counter validar a assinatura do valor público de Diffie-Hellman com uma chave pública, obtida numa das tabelas de configuração, a identidade do interlocutor é o *hash* dessa chave pública, independentemente do que for indicado através do canal. Se um atacante capturar e usar um valor público de Diffie-Hellman assinado, ele poderá personificar temporariamente a entidade a quem o valor foi capturado, mas assim que existir comunicação sobre o canal seguro, com controlo de integridade, imediatamente se detectará que a chave de sessão do mesmo não está correcta.

Curiosamente pode-se observar que o processo de identificação e validação dos intervenientes funciona como um ciclo (ver Figura 18), em que a chave pública é utilizada para a geração do identificador, o valor de *hash* da mesma, sendo essa identidade verificada durante a execução do protocolo Diffie-Hellman com base na assinatura dos valores públicos e confrontação dessa assinatura com a correspondente chave pública. Esta escolha é facilitada devido ao facto do sistema REVS atribuir a cada Counter um par de chaves assimétricas. Dessa forma pode-se garantir que apenas os servidores autorizados podem participar no anel de comunicação e os Counters anterior e seguinte terão de ser dois dos que estão identificados na tabela que possui a identificação dos Counters permitidos.

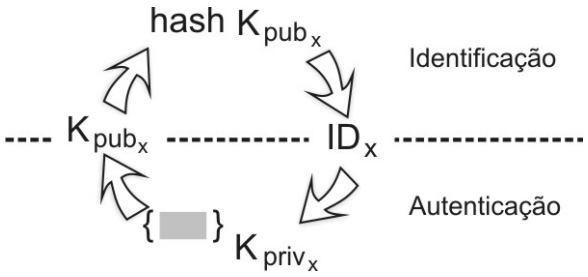

Figura 18 – Ciclo de utilização das chaves assimétricas na identificação / autenticação dos Counters no anel

5.2. Manutenção do RMR

Após o estabelecimento do anel, torna-se necessária a monitorização do estado de funcionamento do mesmo, de forma a garantir a robustez do sistema proposto e

providenciar a tolerância a falhas. Esta operação tem como principal função ajudar a identificar a eventual ocorrência de uma falha na rede e desencadear a execução de medidas que providenciem a recuperação do anel para voltar a um estado de funcionamento normal.

Para alcançar os objectivos anteriormente referidos propõe-se a execução de um protocolo do tipo HELLO, semelhante aos utilizados nos trabalhos analisados. Pretende-se assim que este protocolo seja o responsável pela manutenção das relações entre os vizinhos no anel. Este protocolo poderia servir, também, para verificar a comunicação bidireccional entre a vizinhança, ou seja, um determinado Counter verificaria a operacionalidade tanto dos nós a montante como dos nós a jusante de si. Contudo, para esta implementação não é necessária essa verificação, uma vez que a comunicação pretendida para o anel é unidireccional. Deste modo, atribui-se a responsabilidade de recuperação do anel apenas ao nó anterior e não ao nó seguinte, como veremos mais à frente.

Nos parágrafos seguintes será descrito o protocolo HELLO usado. No entanto, na prática esse protocolo foi embebido no funcionamento normal do anel. Como o anel mantém um tráfego permanente, seja de tráfego útil (votos) ou *cover traffic*, o protocolo HELLO foi embebido em cada mensagem deste tráfego, bastando para tanto que cada mensagem entre Counters tenha uma resposta.

Após a inserção no anel (término das operações de construção do anel), cada um dos Counters inicia um processo de troca de mensagens do tipo HELLO com o nó imediatamente a seguir a si. O envio destas mensagens é realizado periodicamente (cada T_{HELLO} segundos). Esta comunicação tem como principal objectivo verificar o estado em que se encontra o nó seguinte e ao mesmo tempo obter a identidade do Counter adiante do seguinte, pelo que como resposta o nó emissor deverá receber uma mensagem HELLO_ACK(C_{seg}). Com esta resposta o Counter fica informado sobre a operacionalidade do Counter seguinte actual e sobre a identidade do Counter adiante dele no anel.

5.2.1. Determinação de falha no RMR

Considera-se que ocorreu uma falha num Counter **y** quando este não responde às mensagens do tipo HELLO, enviadas para si pelo Counter **x**, durante um período de tempo igual a $T_{MAXHELLO}$ segundos (ver Figura 19).

Assim que **x** detectar a falha de **y** deve comunicar a mesma ao Counter seguinte a **y, z** (recorde-se que a identidade de **z** foi obtida por **x** através de respostas HELLO_ACK enviadas por **y**). Para esse efeito, é utilizada uma mensagem do tipo HOP_FAIL(**y**). O Counter **z** iniciará então o protocolo de junção ao anel, enviando mensagens JOIN_REQUEST para os Counters anteriores, incluindo o Counter **y** que supostamente falhou. Após a obtenção de um novo Counter seguinte por **x**, esta alteração será comunicada ao Counter anterior aquando da execução do protocolo de HELLO iniciado por este último.

O motivo para manter em cada Counter o identificador do Counter adiante do seu seguinte deve-se a razões de desempenho do protocolo de recuperação de falhas. Este valor poderia ser obtido directamente a partir da tabela de encaminhamento utilizada para a iniciação do anel. No entanto, existiria a possibilidade de o mesmo não estar disponível nesse preciso instante, estando a ser usado outro segundo a ordem estabelecida.

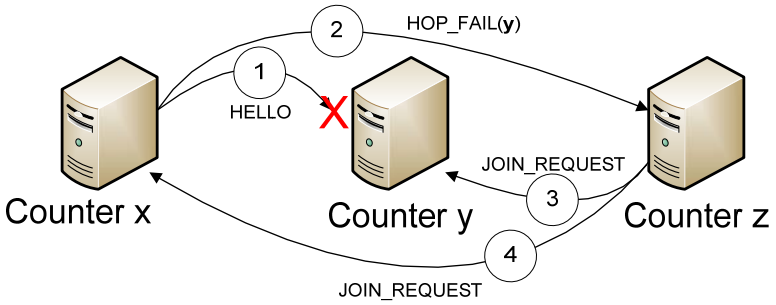

Figura 19 – Descrição do processo de recuperação de uma falha no anel

Nas situações em que a falha ocorre em Counters consecutivos (dois ou mais), o Counter x não consegue obter retorno da invocação remota, HOP_FAIL(**y**), realizada sobre z. Assim, o Counter x irá percorrer sucessivamente a tabela Ord_Tab, a partir do número de ordem do Counter z, até conseguir obter uma resposta à comunicação de HOP_FAIL(**y**). Pelo que o Counter contactado será responsável pela recuperação do anel através dos procedimentos descritos anteriormente para uma única falha.

5.3. Alteração do RMR por intervalos temporais

Como já foi referido anteriormente, para cada período de tempo predefinido pressupõem-se uma ordenação do anel. Assim, torna-se necessária a reconstrução do anel a cada T_{anel} segundos, uma vez que este intervalo de tempo corresponde à duração de uma determinada configuração do anel.

O processo de alteração da estrutura do anel é realizado da mesma forma como se da primeira construção se tratasse. A única preocupação que existe prende-se com a determinação do instante em que tal mudança deve ocorrer. Deste modo, é fundamental a existência de sincronismo dos relógios entre os diversos participantes no anel, só assim poderá ser possível que estas mudanças ocorram de uma forma coerente.

Uma vez que nem todos os Counters iniciam ao mesmo tempo, por exemplo devido à ocorrência de uma falha, é definido um tempo T_{CHG_RING} para cada Counter. Este T_{CHG_RING} representa o tempo que falta desde que o Counter inicia, ou altera a sua configuração, até à hora determinada para fim da configuração em vigor.

Esse tempo é utilizado apenas como um gatilho para accionar o protocolo de construção do anel. Contudo, é necessário ter particular atenção às mensagens a circular no anel, de forma a garantir que as mesmas não são perdidas devido à execução desta alteração no anel. Assim, assume-se a possibilidade de ocorrência de regimes transitórios entre configurações contíguas. Contudo, durante esses períodos de tempo os Counters em migração para a nova configuração deixam de aceitar conexões para troca de mensagens e de verificação do protocolo HELLO. Deste modo, do ponto de vista do Counter anterior, que se pressupõe atrasado na migração para a nova configuração, ocorreu uma falha no seu Counter seguinte (deixou de responder), recuperando o anel até ao momento em que ocorrer a sua transição para a nova ordenação. Note-se que mesmo que este seja o único Counter desfasado no tempo, tal facto não implica uma perda de mensagens, as mensagens em espera nesse Counter são retidas até que o mesmo seja anexado ao anel. A partir desse momento, as mensagens retidas são de novo colocadas em circulação no anel e eventualmente, se for o caso, são processadas.

5.4. Transmissão de mensagens no RMR

Após a construção do anel, este encontra-se preparado para a transmissão de mensagens. As mensagens a circular no anel são cifradas por camadas, utilizando a chave pública de um Counter por camada, independentemente de as mesmas corresponderem a votos ou *cover traffic*. As primeiras são construídas ao nível da aplicação cliente utilizada pelo votante, enquanto que as segundas são criadas pelos diversos Counters de modo a garantir que existe um fluxo de tráfego constante a circular pelo anel. O processo de construção das mensagens será analisado com mais detalhe no próximo capítulo.

Dependendo das condições da rede, poderemos ter situações em que só temos *cover traffic* a circular no anel ou situações em que a quantidade de *cover traffic* em circulação é diminuta. Contudo, a quantidade de tráfego em circulação no sistema permanece constante. Deste modo, é necessário adoptar mecanismos de controlo, em cada Counter, que garantam essa estabilidade do tráfego sem condicionar o funcionamento do sistema.

Como se pode observar na Figura 20, o número de mensagens existente em cada Counter, num determinado momento, depende de três factores: (i) mensagens recebidas do Counter anterior; (ii) mensagens provenientes da submissão de votos por parte da aplicação cliente; (iii) e mensagens de *cover traffic* geradas pelo próprio Counter.

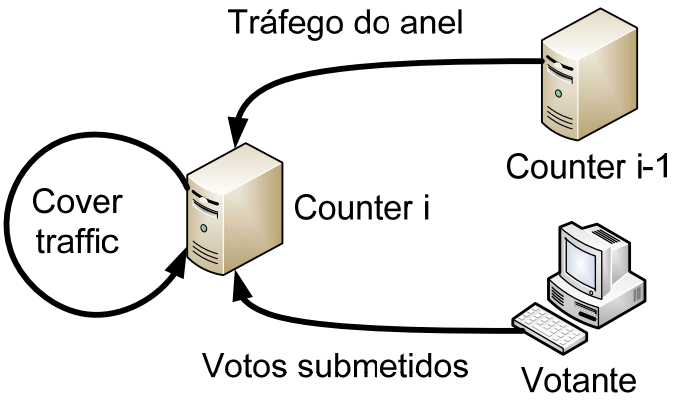

Figura 20 – Identificação da origem das mensagens existentes num Counter

Para a gestão das mensagens cada Counter possui dois *buffers* do tipo FIFO, um de entrada (Be) e outro de saída (Bs). Inicialmente, cada Counter preenche o seu Bs com um número aleatório de mensagens de *cover traffic*. Deste modo, é introduzido um atraso não

determinístico em cada Counter, impedindo a análise temporal das comunicações por parte de observadores externos.

Todas as mensagens recebidas provenientes do Counter anterior são armazenadas no Be e ficarão a aguardar o processamento pelo protocolo de nível superior, responsável pela análise e processamento, se necessário, da mensagem. Aqui poderão ocorrer as seguintes situações: (i) verifica-se que a mensagem, após processamento, corresponde a um recibo, pelo que a mesma é enviada para o cliente que a gerou; (ii) a mensagem circulou ao longo do anel o número máximo de vezes permitido, evidenciando que necessita de processamento por parte de um Counter ausente ou que a mensagem está mal formada. Como tal, essa mensagem é removida do sistema através de mecanismos de remoção de lixo; (iii) se a mensagem recebida não se enquadrar em nenhuma das situações descritas anteriormente, as mensagens poderão eventualmente ser processadas pelo Counter e são colocadas no Bs, onde ficam à espera da sua vez para serem encaminhadas para o Counter seguinte. Se por qualquer motivo o Counter possuir o seu Be completamente preenchido, este recusa a recepção das mensagens provenientes do seu antecessor, o qual deverá manter as mesmas em espera no seu Bs.

Na Figura 21 pode-se visualizar esquematicamente os procedimentos executados aquando da recepção de uma mensagem proveniente do Counter anterior.

No que respeita às mensagens recebidas provenientes da aplicação cliente, estas passam imediatamente para o Bs. Estas mensagens estão directamente ligadas ao objectivo principal do sistema, impondo a cada Counter uma gestão dos seus Be e Bs baseada numa política de compromisso que garanta, em condições normais, a aceitação de todas as mensagens de voto submetidas para esse Counter. No entanto, caso o Counter não possa aceitar a mensagem, por congestionamento na rede (identificado pela falta de espaço no Bs), essa informação é remetida para o Cliente que efectuou a submissão indicando que o mesmo deverá tentar contactar outro Counter para proceder à submissão do seu voto.

As mensagens de *cover traffic* são geradas pelos Counters durante todo o período de operação do sistema. Inicialmente, como já foi referido anteriormente, aquando do arranque o Counter gera um número aleatório de mensagens do tipo *cover traffic*. Estas mensagens têm a particularidade de possuírem como destino final o mesmo Counter que as criou, permitindo assim a sua substituição por outras no momento da sua recepção.

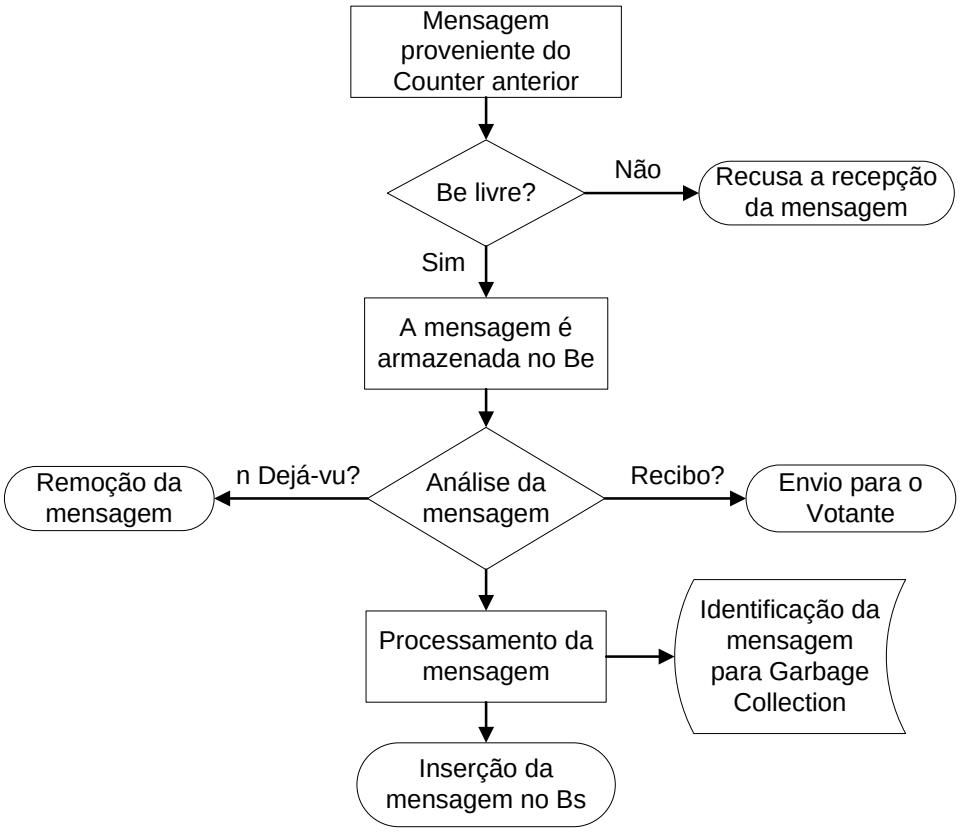

Figura 21 – Diagrama de procedimento a executar após a recepção de uma mensagem proveniente do Counter anterior

Contudo, uma vez que as condições de acesso ao sistema poderão variar ao longo do tempo, torna-se necessário implementar um sistema de controlo para as mensagens de *cover traffic*, de forma a garantir que a existência das mesmas não impeça a transmissão das mensagens contendo os votos submetidos. Para isso, deve-se impor um limiar no Bs, de modo a que se esse limiar for ultrapassado, o Counter deixa, temporariamente, de produzir novas mensagens de *cover traffic* para a rede.

As mensagens são reencaminhadas para o Counter seguinte através da invocação de métodos remotos e apenas são removidas do Bs quando se verifica que a invocação remota foi executada com sucesso, correspondendo à passagem da mensagem para o Counter seguinte. Com a utilização de RMI para a troca de mensagens, implementa-se um sistema de confirmação automático baseado no valor de retorno da invocação. Garante-se também a conservação das mensagens para as quais não seja possível executar a transmissão para o Counter seguinte.

Na prática, uma vez que o Bs é implementado segundo uma disciplina do tipo FIFO, a conservação das mensagens é realizada através de dois processos distintos (i) cópia e (ii) eliminação. Assim, o que é efectivamente enviado para o Counter seguinte corresponde a uma cópia da mensagem armazenada no Bs, sendo a mensagem original eliminada apenas após a confirmação da transmissão. Deste modo, garante-se, para além da conservação das mensagens, o respeito pela ordenação das mensagens tendo em conta a ordem de chegada. Este facto revelar-se-á de enorme importância aquando da discussão dos mecanismos de remoção de lixo.

Contudo, se ocorrer uma falha num Counter todas as mensagens presentes unicamente nos seus buffers serão perdidas, em virtude de não existir armazenamento de mensagens em memória secundária.

Na figura seguinte pode-se visualizar uma caracterização de um esquema representativo do modelo de gestão de mensagens implementado para cada Counter. Neste esquema, pode-se facilmente identificar as três fontes de mensagens no sistema, evidenciando-se o processamento a que estão sujeitas todas as mensagens provenientes do Counter anterior. A ordem pela qual são apresentadas as fontes é representativa da gestão de mensagens ao nível do Counter, apenas são geradas novas mensagens de *cover trafiic*, para substituir as recebidas, se o Bs estiver abaixo do limiar definido para a eleição, seguidamente as mensagem de voto têm prioridade relativamente às procedentes do Counter anterior, uma vez que estas últimas poderão aguardar a sua vez no Be.

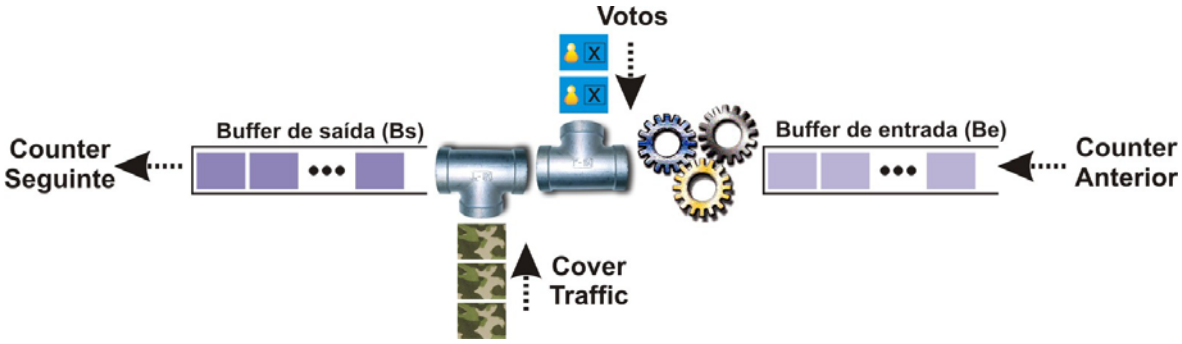

Figura 22 – Representação do processo de gestão de mensagens num Counter

5.5. Remoção de lixo (Garbage Collection)

Pelos motivos mais variados, desde mensagens mal formadas a mensagens que não podem ser processadas por falha do Counter, podem ocorrer situações de circulação infinita de mensagens. Dependendo da quantidade de mensagens nestas situações, o sistema pode ser globalmente afectado, pelo que torna-se preponderante a definição de estratégias que contrariem a circulação infinita de mensagens no anel.

A solução para este fenómeno é fornecida pela implementação de mecanismos de remoção de lixo (*garbage collection*), sendo que neste contexto é considerado lixo todas as mensagens que depois de terem circulado um número finito de vezes no anel não puderam ser processadas. Nestas circunstâncias podem encontrar-se os seguintes tipos de mensagens:

1. Mensagens contendo votos, em que pelo menos um dos Counters, pertencente ao LMR escolhido, não se encontra activo;
2. Mensagens de *cover traffic*, enviadas para processamento por Counters que não se encontram activos ou nas situações em que ocorre uma falha nos Counters emissores após o envio;
3. Mensagens mal formadas, enquadram-se nesta situação todas as mensagens que não podem ser processadas em virtude de o seu conteúdo não poder ser interpretado por nenhum dos Counters autorizados para a eleição.

Para realizar esta tarefa, será utilizado um mecanismo de “dejá vu” das mensagens, ou seja todas as mensagens que passam por um Counter são monitorizadas e contabilizadas, sendo esta informação armazenada numa tabela dinâmica na qual será criada uma nova entrada por cada nova mensagem processada pelo Counter. Nesta situação entende-se por nova mensagem uma mensagem cujo hash seja diferente de todas as existentes na tabela. Ou seja, um Counter X regista a recepção de uma mensagem, embora durante o seu percurso ao longo do anel até ao seu regresso ao Counter X a mesma mensagem foi alterada, deste modo para este efeito ela será vista como uma nova mensagem, sem qualquer correlação com os registos anteriores que possam existir para a mesma.

Contudo, uma vez que a mensagem só deve ser removida após ter sido vista n vezes, cada Counter deverá actualizar a sua tabela sempre que a mesma mensagem for visualizada, em que a actualização da tabela corresponde ao incremento do contador referente a essa mensagem. Este requisito cria a necessidade de introdução de um mecanismo de indexação, que facilite a pesquisa ao longo dos registos existentes na tabela. Assim, a identificação das mensagens será efectuada através da utilização do respectivo valor de hash das mesmas.

Com a resolução do problema de circulação infinita de mensagens pode conduzir o sistema a novos problemas similares, como é o caso do crescimento infinito da tabela de controlo das mensagens. No entanto, este último é de resolução mais simples do que o primeiro. Como vimos anteriormente, o processamento de mensagens segue uma disciplina do tipo FIFO, ou seja todas as mensagens são processadas sequencialmente. Assim, sempre que seja detectada uma mensagem que necessite de ser removida do sistema é também removida a sua entrada da tabela e ao mesmo tempo todas as entradas existentes mais antigas que esta. Daí a importância da manutenção da ordenação das mensagens no buffer de saída dos Counters, só sendo removidas as mensagens assim que for estabelecida a comunicação com o Counter seguinte.

Na Figura 23 pode ser observado o diagrama representativo das operações identificadas acima.

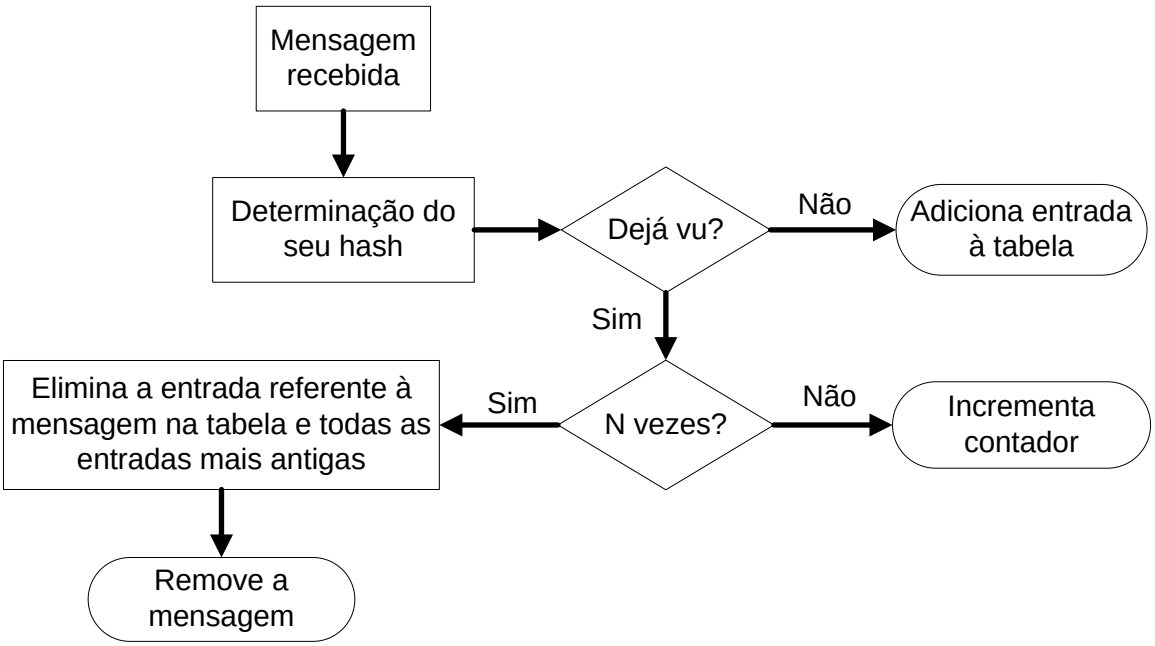

Figura 23 – Diagrama de procedimentos a executar durante a execução dos mecanismos de remoção de lixo (*garbage collection*)

Capítulo 6

Logical Mix Ring (LMR)

Ao contrário do RMR, o qual é composto por todos os Counters autorizados para a eleição, o LMR apenas é constituído por um subconjunto dos Counters escolhidos aleatoriamente pela aplicação cliente do votante ou, no caso de mensagens de *cover traffic*, pelos Counters associados à mensagem aquando da sua construção. Deste modo, pode-se ver o RMR como a rede de suporte à execução do LMR, podendo este último ser visto como um protocolo de camada superior.

Nas secções seguintes serão apresentadas as funcionalidades executadas ao nível do LMR, as quais poderão ser resumidas à construção e processamento de mensagens. No entanto, as mensagens podem ser construídas pela aplicação cliente ou por um Counter, consoante sejam mensagem contendo votos ou *cover traffic* respectivamente.

6.1. Construção das mensagens

Todas as mensagens em circulação no anel possuem a mesma estrutura de dados e consequentemente o seu comprimento é o mesmo independentemente do tipo de mensagem. Este facto deve-se a um dos objectivos do sistema que é esconder o tráfego entre dois pontos, deste modo para que o tráfego real não possa ser distinguido

relativamente ao *cover traffic*, é preponderante que ambos pareçam equivalentes aos olhos de um observador externo. No entanto, em termos de conteúdo e de construção existem diferenças significativas.

O processo de construção as mensagens, salvo as condicionantes apresentadas nas secções seguintes, são similares para ambos os tipos de mensagens, podendo ser visualizado na figura seguinte. Assim, teremos que em todas as situações o processo de construção da mensagem segue o percurso inverso relativamente ao sentido de propagação da mesma. Sendo a mensagem construída por camadas de cifras com as chaves públicas dos respectivos destinatários intermédios.

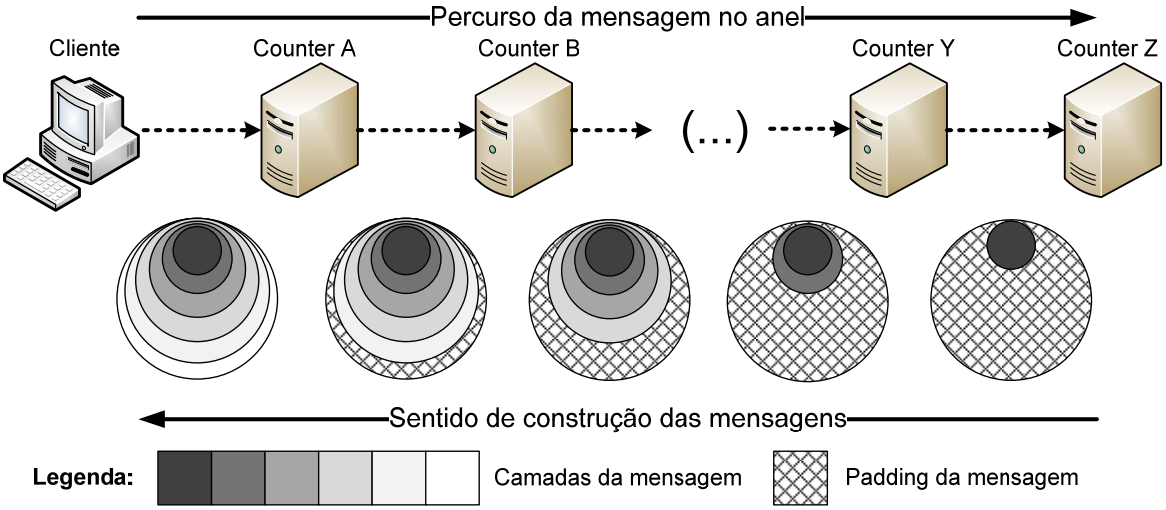

Figura 24 - Esquema simplificado do processo de construção e transmissão das mensagens

6.1.1. Mensagens de submissão de votos

Este tipo de mensagens são construídas pela aplicação cliente do REVS, contendo duas informações importantes para o sistema – o voto e o recibo para o cliente. O recibo aqui mencionado encontra-se descrito em [5] e corresponde em termos práticos a um *byte array* gerado aleatoriamente pela aplicação, não possuindo qualquer tipo de correlação entre seu valor e a escolha efectuada pelo votante.

O primeiro passo a ser executado prende-se com a selecção dos Counters envolvidos no processamento da mensagem. Esta escolha deverá ser aleatória correspondendo a um subconjunto dos Counters, autorizados para a eleição, designado por

LMR em virtude de serem os nós responsáveis pela decifra da mensagem aquando da sua circulação no anel.

Após a determinação dos Counters responsáveis pelo processamento da mensagem, iniciam-se os procedimentos efectivos de construção da mensagem. Tal como vimos anteriormente, este processo segue o sentido inverso da recepção. Assim, a construção será iniciada pelo recibo que, juntamente com o endereço IP do cliente, será cifrado com a chave pública do último Counter escolhido para pertencer ao LMR. Juntamente com a informação anterior, esta parte da mensagem será marcada com a marca (*tag*) identificativa do tipo de mensagem, neste caso identificando-a como uma mensagem contendo um recibo que deverá ser devolvido ao cliente através de um processo de *fan-out* externo. Ver figura seguinte.

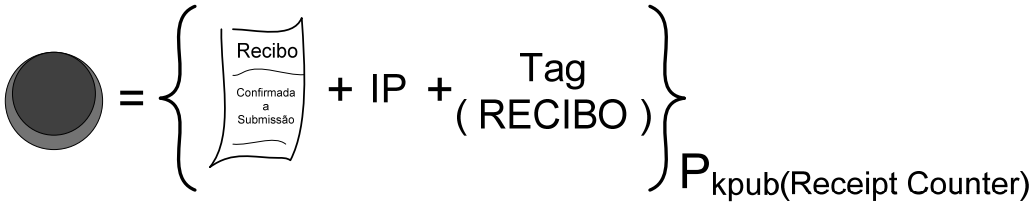

Figura 25 - Construção da primeira camada da mensagem, contendo o recibo

Seguidamente, ver Figura 26, a mensagem sofre pelo menos mais uma cifra relativa a um ou mais Counters intermédios, responsáveis pela separação entre o Storage Counter e o Receipt Counter. Estas cifras utilizam as chaves públicas dos respectivos Counters e são identificadas com a *tag* informativa de reenvio para a rede, indicando ao receptor que não necessita de realizar mais nenhuma operação sobre a mensagem para além da decifra.

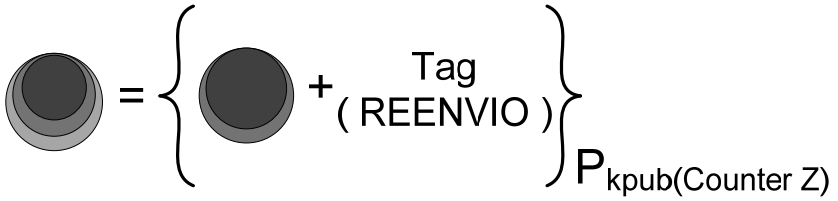

Figura 26 – Composição das camadas que medeiam o Storage Counter e o Receipt Counter

Chega-se ao momento de inserção do voto na mensagem. A aplicação cliente deve juntar à mensagem, construída até ao momento, o boletim de voto preenchido e assinado

pelas autoridades da eleição. O conjunto é então cifrado com a chave pública do Counter escolhido para actuar como *Storage Counter*, sendo anexada à mensagem a *tag* indicativa da necessite de proceder à operação de *fan-out* interna para armazenamento do voto, como pode ser visualizado na Figura 27.

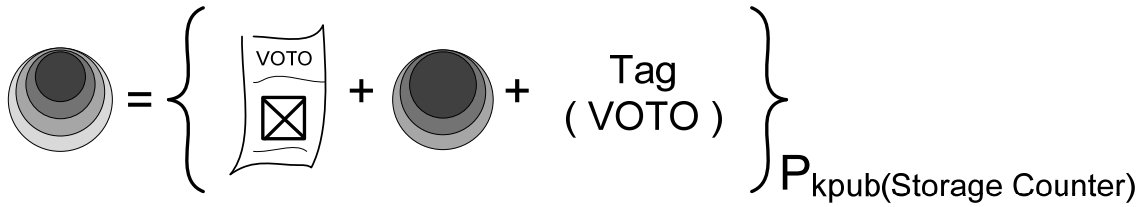

Figura 27 – Composição da camada portadora do Voto e destinada ao Storage Counter

Por fim, como se pode observar na Figura 28, são inseridas uma ou mais cifras à mensagem com o intuito de esconder o *Storage Counter* do *Submission Counter*. Por cada camada adicionada será adicionada a respectiva *tag* de reenvio da mensagem e todo o conteúdo cifrado com a chave pública dos respectivos Counters. Finaliza-se assim o processo de construção das mensagens, estando a mesma preparada para ser submetida.

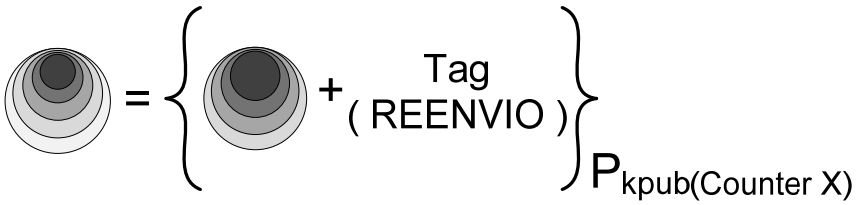

Figura 28 - Composição das camadas pertencentes à mensagem e que antecedem a recepção da mesma pelo Storage Counter

O último passo da aplicação cliente será seleccionar um dos Counters disponíveis de modo a submeter a sua mensagem para o anel através do mesmo. Evidentemente, com o processo de submissão do voto será efectuado através uma invocação de métodos remotos, se o Counter escolhido estiver inoperacional será lançada uma excepção do Java RMI, pelo que a aplicação deve escolher um novo Counter para proceder à submissão desta mensagem.

Eventualmente a aplicação cliente poderá não utilizar o número máximo de Counters definido para o processamento, necessitando nestes casos de proceder ao preenchimento (*padding*) da mesma até obter uma mensagem com o comprimento definido

para a eleição. De modo a ocultar o tráfego em circulação na rede, esta é uma operação primordial, sendo o tamanho da mensagem proporcional ao número de nós utilizado.

6.1.2. Mensagens de cover traffic

Sendo o processo de construção semelhante ao apresentado na secção anterior, existem contudo diferenças significativas. Este tipo de mensagens têm como principal função camuflar as mensagens submetidas pela aplicação cliente, daí a sua designação *cover traffic*, sendo construídas pelos Counters existentes na rede. As mensagens de *cover traffic* são construídas na ocorrência dos seguintes eventos:

1. Arranque do Counter, nesta situação é gerada uma quantidade aleatória de mensagens, introduzindo um atraso por nó não determinístico;
2. Aquando da recepção de uma mensagem de *cover traffic* gerada por si.

Relativamente às mensagens analisadas anteriormente, estas têm a particularidade de não necessitarem de recorrer às *tags* relativas ao voto e recibo em virtude das mesmas não conterem conteúdos válidos. Deste modo, para a construção da mensagem de *cover traffic* apenas será necessário utilizar, como destinatário intermédio, um Counter que não o criador da mensagem e escolhido aleatoriamente, identificando a mensagem com a *tag* de reenvio. Note-se que para que o Counter possa substituir a mensagem por outra no momento da sua recepção, necessita de se definir a si próprio como o destinatário final da mensagem, utilizando uma *tag* denominada cover para identificação da mesma no momento da recepção.

Contudo, necessita-se que esta mensagem, aos olhos de um observador externo (aqui por externo pretende-se dizer outro observador que não o construtor da mensagem), não possa ser distinguida de outra mensagem, nomeadamente das que contenham votos. Para atingir este objectivo necessita-se que todas as mensagens em circulação no anel possuam o mesmo tamanho e estrutura, sendo o tamanho alcançado através dos procedimentos de *padding* da mensagem. Na figura seguinte pode ser observado o esquema de construção das mensagens de *cover traffic*. Sendo o Counter X o Counter

responsável pela construção da mensagem e o Counter Y o Counter escolhido para o processamento intermédio da mensagem.

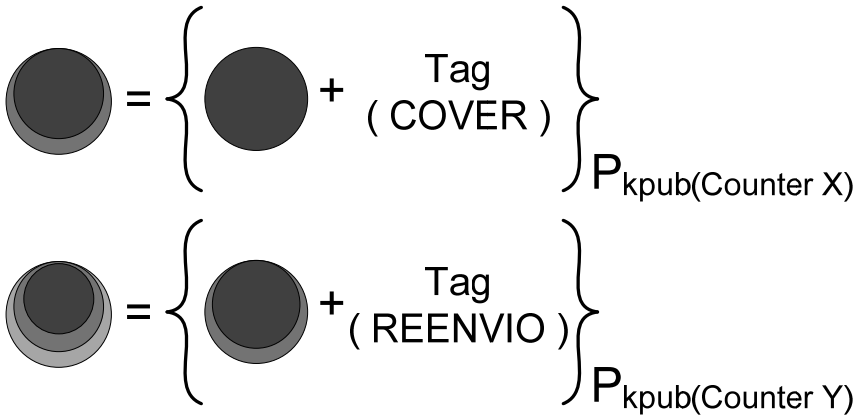

Figura 29 - Procedimentos para a construção de mensagens de cover *traffic*

As mensagens de *cover traffic*, como se pode verificar pela demonstração do processo de construção, possuem apenas duas camadas cifradas, pelo que são de construção menos complexa.

6.2. Processamento de mensagens

No que se refere ao processamento das mensagens em circulação no anel este é indistinto para ambos os tipos de mensagens, tal como seria de prever. Para cada Counter será impossível distinguir uma mensagem de *cover traffic* de outra contendo um voto, excepto se essa mensagem corresponder a uma das operações especiais – submissão do voto, armazenamento de voto, devolução do recibo. Ou seja, todas as mensagens recebidas e identificadas com a *tag* reenvio possuem a mesma probabilidade de transportar votos.

O processamento das mensagens resume-se à verificação e validação do destinatário, caso o Counter seja o destinatário da mensagem esta é decifrada e analisada a *tag* para proceder à operação definida pela mesma. Por cada processamento da mensagem, o Counter deverá realizar as operações de *padding* necessárias que permitam à mensagem continuar a sua viagem pelo anel com o tamanho constante. Na Figura 30 pode ser observado o esquema representativo do processamento de uma mensagem contendo um voto ao longo do anel. Apesar de na figura encontrarem-se todas as operações realizadas

num único percurso ao longo do anel, este não é um requisito necessário e até muito pouco provável. Em condições normais, com todos os Counters presentes, o número máximo de voltas apenas é limitado pelo número máximo de Counters escolhidos para processar a mensagem, sendo este limite igual ao número de Counters.

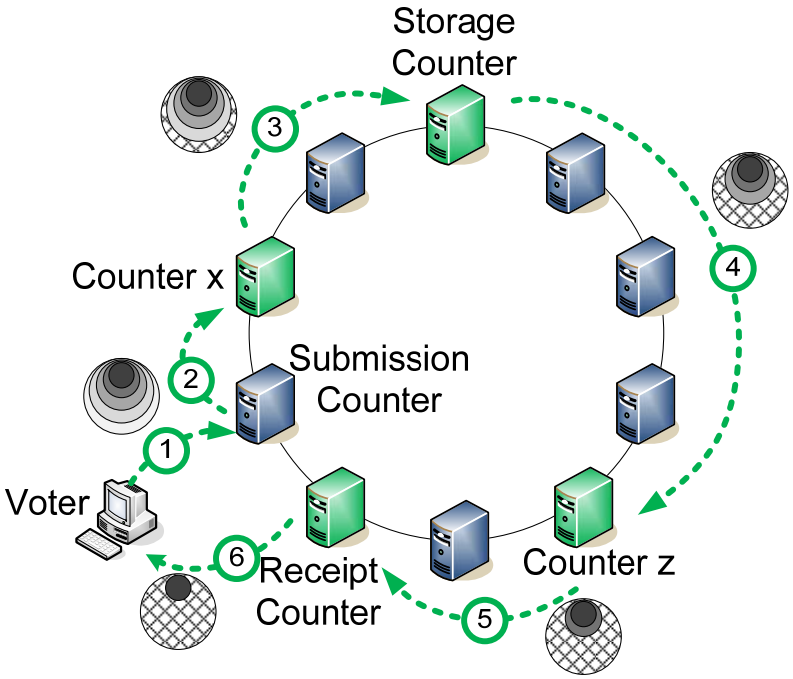

Figura 30 – Fases de processamento de uma mensagem portadora de um voto e consequentemente submetida por um cliente

O processamento das mensagens de *cover traffic*, é mais simples em virtude do número diminuto de camadas utilizadas, contudo é necessário preencher a mensagem de modo a garantir o tamanho uniforme utilizado no anel. O esquema de processamento das mensagens ao longo do anel pode ser observado na figura seguinte.

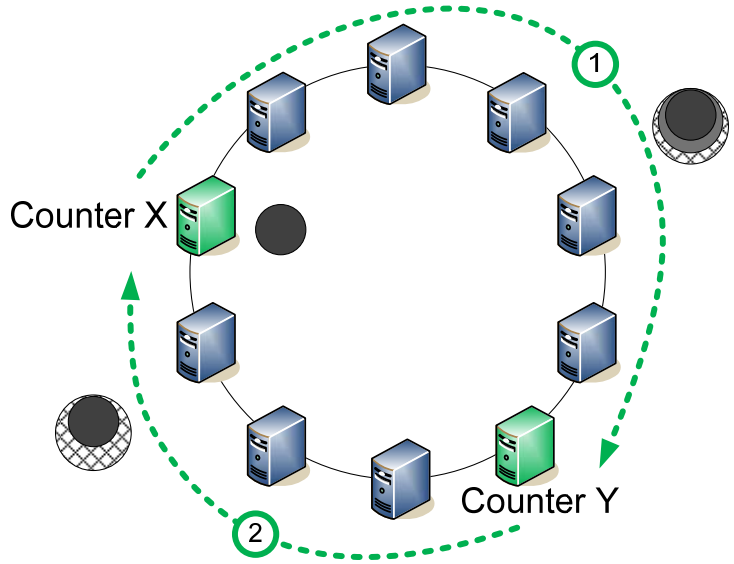

Figura 31 – Processamento de uma mensagem de *cover traffic*

6.3. Cover traffic como coração do anel

Seguindo o conceito inicial dos Mix Rings em que as mensagens de *cover traffic* funcionam como um batimento cardíaco que anuncia o estado do anel, pode-se utilizar o mesmo conceito nesta nova arquitectura com o objectivo de informar o cliente sobre o estado, num dado instante, de cada Counter no anel.

Como vimos anteriormente, cada Counter, no arranque do sistema, gera um número aleatório de mensagens do tipo *cover traffic*. Aquando da recepção, essas mensagens serão substituídas por outras mensagens do mesmo tipo, dependendo das condições da rede. Tendo em conta que estas mensagens são construídas de modo a durante uma viagem ao longo do anel ocorrer apenas um processamento por parte de outro Counter, escolhido pelo construtor da mensagem, sendo posteriormente devolvidas à origem.

Assumindo que cada Counter possui a mesma probabilidade de ser escolhido para processar uma mensagem de *cover traffic*, pode-se definir uma estratégia para fornecer a informação aprendida ao cliente após solicitação.

6.3.1. Definição do modelo

As mensagens de cover traffic são construídas de modo a serem processadas por um Counter escolhido aleatoriamente. Sendo a sua principal função camuflar o tráfego em circulação no anel, estas mensagens podem servir como um bom indicador do estado dos diversos Counters autorizados.Podem ocorrer três situações distintas com as mensagens de *cover traffic*:

1. A mensagem é devolvida ao Counter que a gerou após processamento, indicando que o Counter escolhido encontra-se activo;
2. A mensagem é removida, no Counter que a gerou, através do mecanismo de remoção de lixo. Este facto indica que o Counter seleccionado para o processamento da mensagem não se encontra activo;
3. A mensagem é perdida durante a circulação no anel e detectada a perda pela ausência da devolução. Este facto leva a uma situação indefinida, onde nada pode ser deduzido quanto ao estado de funcionamento do Counter em causa.

Cada Counter necessita de manter duas tabelas, uma com informações que correlacionam a mensagem de *cover traffic* produzida com o Counter escolhido e outra onde são armazenados os estados dos vários Counters autorizados a participar no anel.

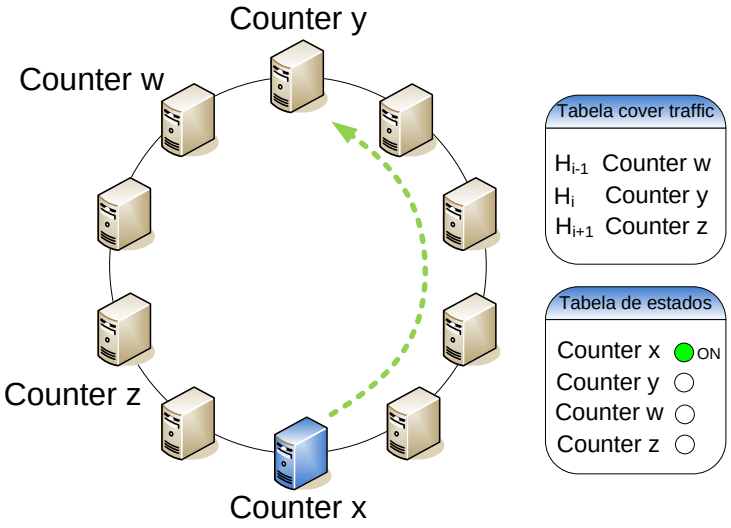

Figura 32 – Inicialização do procedimento de detecção do estado do anel

Após a escolha do Counter intermédio, o Counter necessita de gerar o conteúdo aleatório para a mensagem o qual deverá ser posteriormente cifrado em primeiro lugar com a chave pública do próprio Counter e seguidamente com a chave pública do Counter escolhido.

Para que se possa determinar e monitorizar o estado de operação dos vários Counters, o sistema tem de proceder ao registo da relação entre a mensagem enviada e o Counter escolhido. Para identificação da mensagem será utilizado um mecanismo semelhante ao utilizado no RMR nas operações de remoção de lixo.

Na Figura 32 podemos observar esquematicamente o processo inicial de execução desta operação de monitorização do estado dos Counters no anel. Na tabela "cover traffic" é apresentada a identificação das mensagens produzidas através do respectivo hash e relacionadas com o identificador do Counter de destino. No que toca à tabela de estados, apenas é apresentado, nesse momento, como activo o registo correspondente ao próprio Counter.

Assim, o Counter aquando da recepção da mensagem de *cover traffic* gerada por si, deve proceder à verificação, através do valor do hash guardado previamente, estabelecendo a relação entre a mensagem recebida e o Counter escolhido para processamento da mesma. Após esta consulta deverá ser registado o estado do Counter como online (ver figura seguinte).

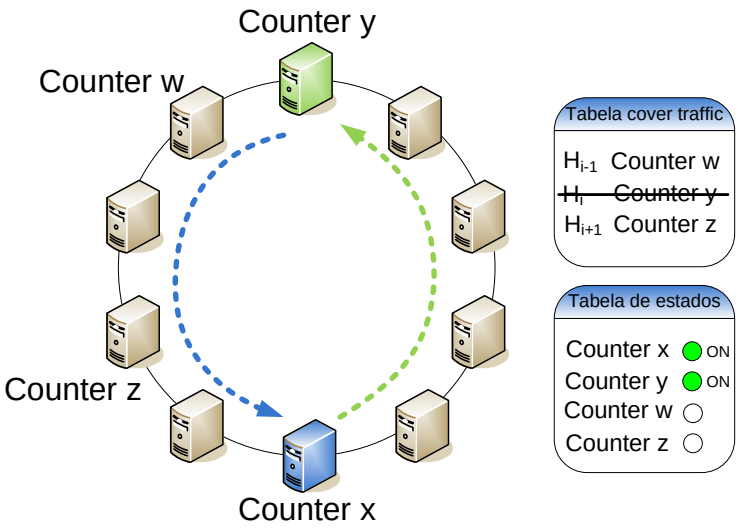

Figura 33 – Apresentação das operações a realizar aquando da recepção de uma mensagem de *cover traffic*

A determinação dos Counters offline é processada de maneira similar à anterior. Contudo, o estado do Counter é actualizado aquando da remoção da mensagem pelo mecanismo de remoção de lixo. Ao mesmo tempo a mensagem é removida do anel e o estado do Counter actualizado, deve-se proceder à limpeza dos registos da tabela de envios, assim caso existam registos de mensagens anteriores à removida em espera e uma vez que a transmissão é efectuada sequencialmente numa disciplina do tipo FIFO, todos esses registos podem ser eliminados sem se concluir qual o estado de operação dos Counter em causa. Este caso pode ser observado na Figura 34.

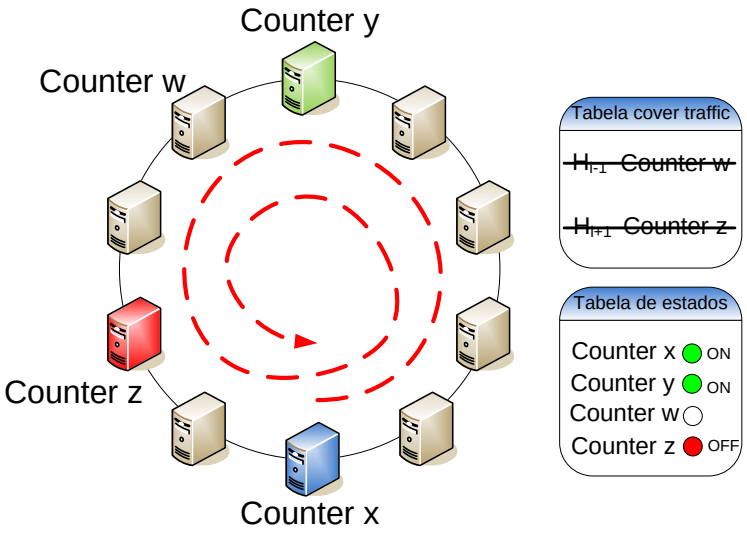

Figura 34 – Operações realizadas sobre as tabelas aquando da remoção de uma mensagem de *cover traffic* por mecanismos de remoção de lixo

6.3.2. Optimizações e segurança do lado do cliente

De modo a garantir alguma qualidade de serviço no acesso ao sistema, a primeira submissão do voto pela aplicação cliente é efectuada sem o conhecimento do estado da rede. Com isto, pretende-se aumentar a eficiência do sistema, bem como proteger o sistema de ataques por parte de Counters maliciosos que através da difusão de informações falsas conseguissem limitar a escolha por parte da aplicação cliente ao conjunto de Counters pretendido, possivelmente a actuar em conluio.

Assim, define-se que apenas após ter sido detectada uma falha na submissão (não recepção do recibo por parte do cliente em tempo útil) é que é permitida à aplicação

inquirir um Counter aleatoriamente, não necessariamente o *Submission Counter*, sobre o estado do anel. Sendo que, a resposta obtida permite, à aplicação cliente, reduzir o leque de Counters a considerar para a construção da mensagem contendo o voto, correspondendo eventualmente a um subconjunto do conjunto considerado inicialmente.

Apesar das informações fornecidas pelo Counter contactado poderem estar desactualizadas, elas servem perfeitamente como um bom ponto de partida para potenciar o sucesso na execução do processo de submissão de votos após a detecção de uma falha.

Capítulo 7

Implementação e validação

O modelo para construção do RMR apresentado anteriormente foi implementado utilizando a linguagem de programação Java, sendo a comunicação entre os vários Counters realizada sobre Java *Remote Method Invocation* (RMI). Deste modo, como já foi referido, as trocas de mensagens anteriormente descritas deixam de ser vistas segundo a relação mensagem – resposta e passam a ser vistas como invocação – retorno (ver Figura 35).

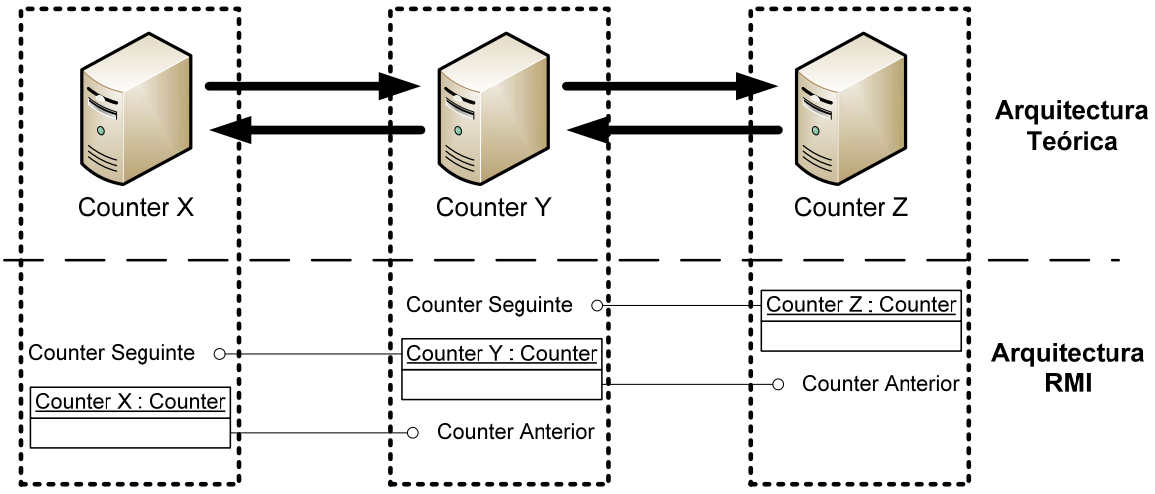

Figura 35 – Comparação do modelo teórico com o modelo implementado

Do mesmo modo, as operações de gestão e manutenção do anel são efectuadas através da captura de excepções lançadas pelo RMI, provenientes da invocação de métodos remotos pertencente a objectos não activos. Grande parte das tarefas necessárias para o correcto funcionamento do anel baseiam-se neste mecanismo de controlo, seja para a: (i) inserção de Counters no anel; (ii) verificação do estado do Counter seguinte; (iii) recuperação do estado de funcionamento do anel; (iv) gestão do encaminhamento de mensagens.

Considerando os pressupostos anteriores para esta implementação, ao contrário dos protocolos de comunicação estudados, o protocolo Hello encontra-se embebido no protocolo de transmissão de mensagens. Considera-se que existindo uma comunicação síncrona regular com confirmação entre os vários Counters pertencentes ao anel, não existe a necessidade de introduzir uma nova comunicação similar para o controlo do estado dos participantes. Deste modo, os passos ii e iv enunciados acima correspondem na prática a uma única operação. Ao mesmo tempo que o Counter estabelece comunicação com o seu Counter seguinte, encontra-se também a efectuar a verificação do estado do mesmo. Para completar esta junção protocolar, apenas é necessário que o retorno da invocação remota para envio de uma mensagem seja a identificação do Counter a seguir ao destinatário, garantindo assim a correcção relativamente ao modelo definido.

7.1. Definição da estrutura de implementação

O sistema aqui apresentado foi idealizado para operar segundo numa perspectiva de sistema distribuído em que o objecto do sistema deverá ser processado por diversos intervenientes até se atingir o resultado pretendido – a submissão anónima do voto com confirmação. Este sistema poderá ser analogamente comparado com um sistema cliente servidor em cascata, uma vez que do ponto de vista de um Counter, este desempenha simultaneamente os papéis de servidor e cliente durante o tempo de vida do anel. Sendo que actua como cliente nas seguintes situações:

- Pedidos de inserção no anel;
- Envio de mensagens;
- Comunicação da ocorrência de falha no anel;

- Devolução do recibo à aplicação cliente.

Na Figura 36 pode-se analisar o diagrama de objectos relativo à implementação do modelo descrito nos capítulos anteriores. Este diagrama corresponde ao definido para uma entidade do tipo Counter quando desempenha o papel de cliente.

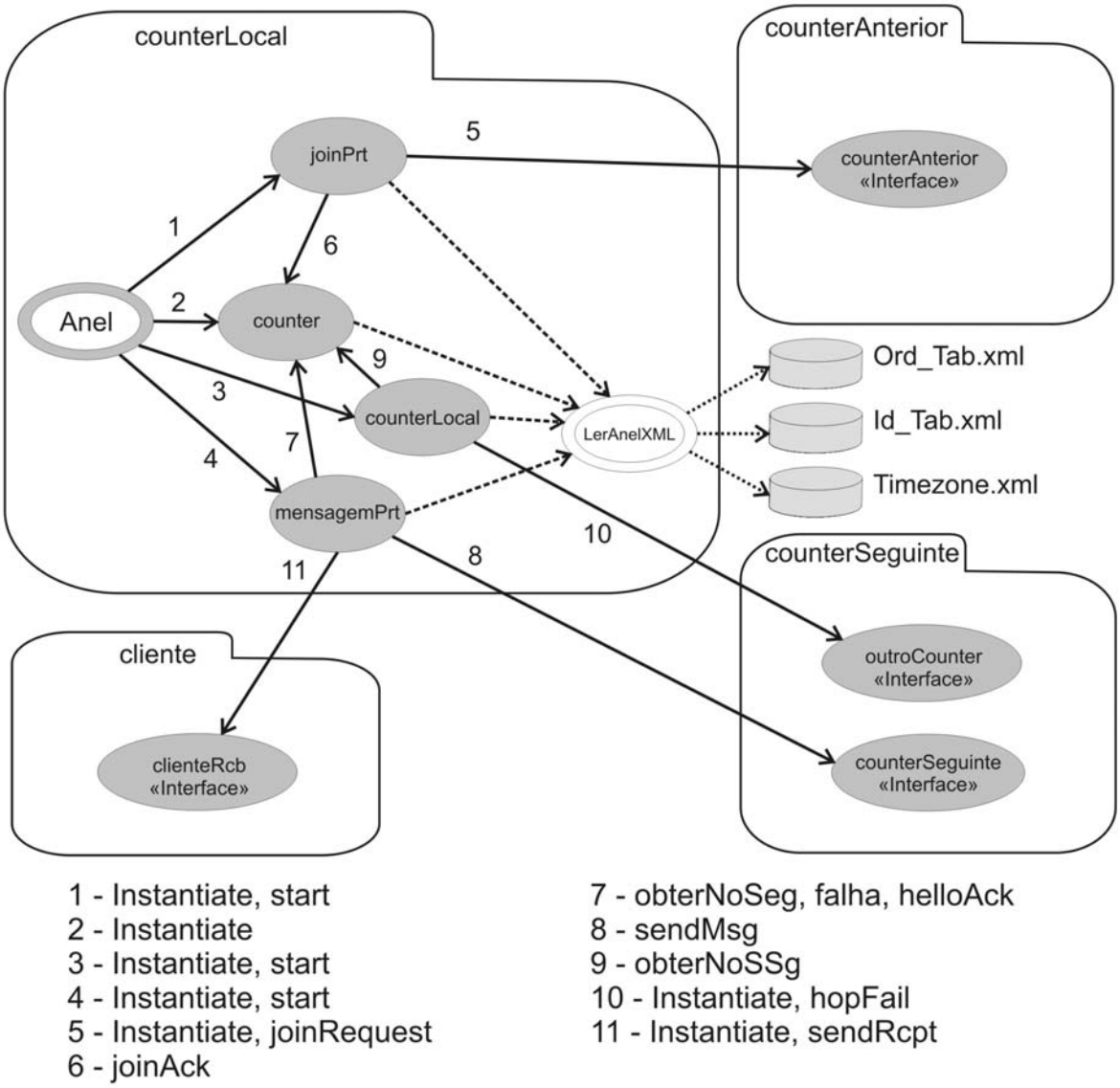

Figura 36 - Diagrama de objectos para a implementação de um Counter pertencente ao anel a desempenhar o papel de cliente

No entanto, existem outras situações onde o mesmo Counter actua como um servidor, respondendo às solicitações recebidas quer de outros Counters, quer das

aplicações clientes aquando da submissão de votos. Assim, as situações identificadas em que um Counter executa operações de servidor ocorrem quando:

- Recebem pedidos de inserção no anel por parte de um Counter posterior;
- Recebem mensagens provenientes de um Counter anterior;
- Recebem uma mensagem contendo um voto proveniente da aplicação cliente;
- Recebem um aviso de ruptura do anel.

Deste modo, na Figura 37 encontra-se uma representação para o diagrama de objectos da entidade Counter quando desempenha o papel de servidor. As implementações do objecto remoto poderiam estar todas concentradas na mesma interface, contudo por uma questão de clareza optou-se por separar as interfaces consoante as mesmas são contactadas por Counters anteriores, posteriores ou votantes.

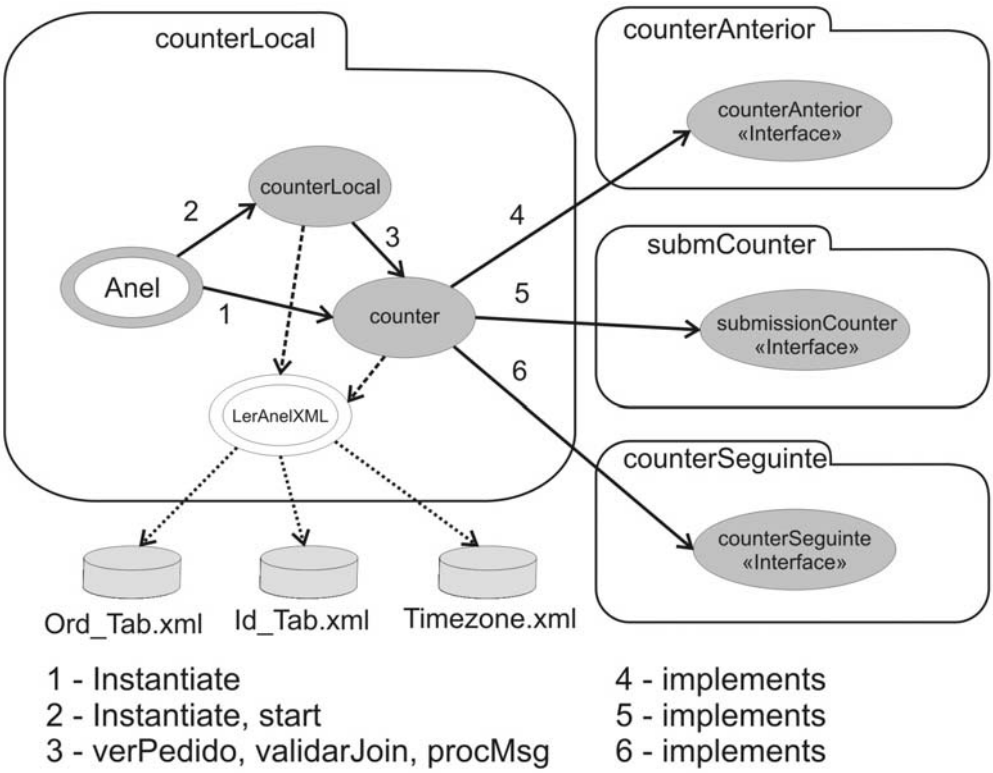

Figura 37 - Diagrama de objectos para a implementação de um Counter pertencente ao anel a desempenhar o papel de servidor

Como foi referido, nem só os Counters desempenham o papel de clientes, existindo uma situação em que a solicitação surge por parte do votante, pelo que na Figura 38 apresenta-se o diagrama de objectos para este caso.

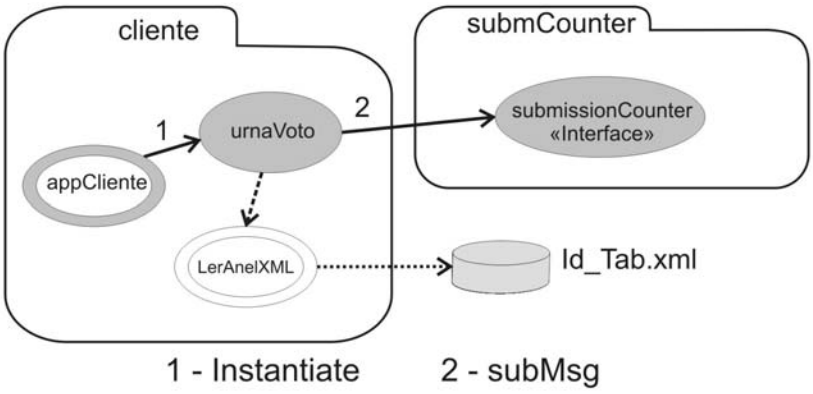

Figura 38 - Diagrama de objectos para a implementação da aplicação cliente

7.2. Identificação das zonas críticas

Com base nos diagramas apresentados na secção anterior e na modelação do sistema exposto no Capítulo 5 , verifica-se que este sistema, sendo distribuído, possuirá zonas críticas de acesso a dados.

Para a gestão do acesso a essas zonas, o sistema recorre à utilização de semáforos de modo a garantir a exclusividade no acesso e consequentemente garantindo a consistência da informação.

Cada Counter guarda informações relativas à sua vizinhança. No entanto, em virtude do protocolo de gestão do anel ser convergente, no sentido de que para um dado Counter se o seu Counter anterior não corresponder ao definido pela eleição para esse intervalo temporal, este continuará na busca da configuração original. Assim, torna-se necessário criar mecanismos que impeçam um Counter de efectuar esses procedimentos de convergência e ao mesmo tempo aceite pedidos de inserção no anel.

Na figura seguinte pode ser observado o esquema representando o ciclo de vida do protocolo de inserção no anel (*Join*).

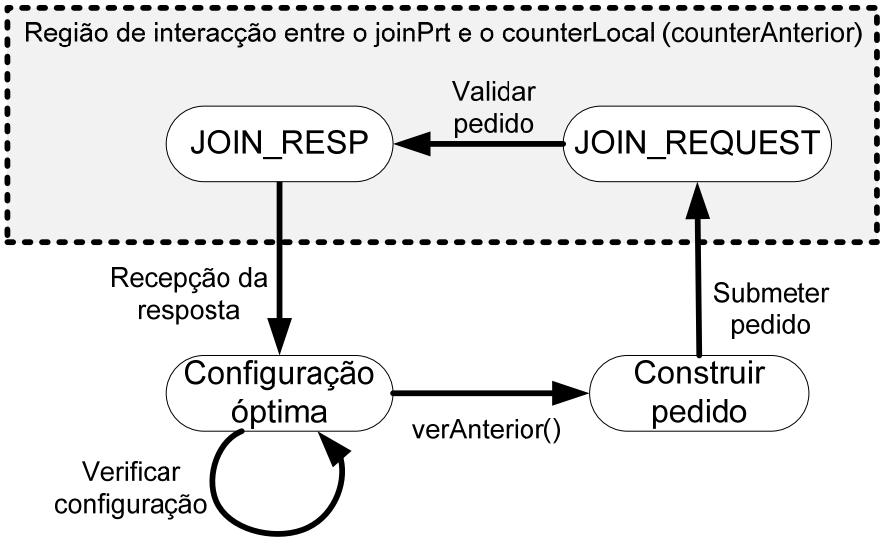

Figura 39 - Ciclo de vida da thread joinPrt, responsável pela inserção e convergência do anel

Similarmente, em virtude do protocolo de verificação do Counter seguinte se encontrar fundido com o protocolo de envio de mensagens, a thread mensagemPrt irá possuir uma região crítica no momento em que submete a sua mensagem para o Counter seguinte, como se pode visualizar na Figura 40.

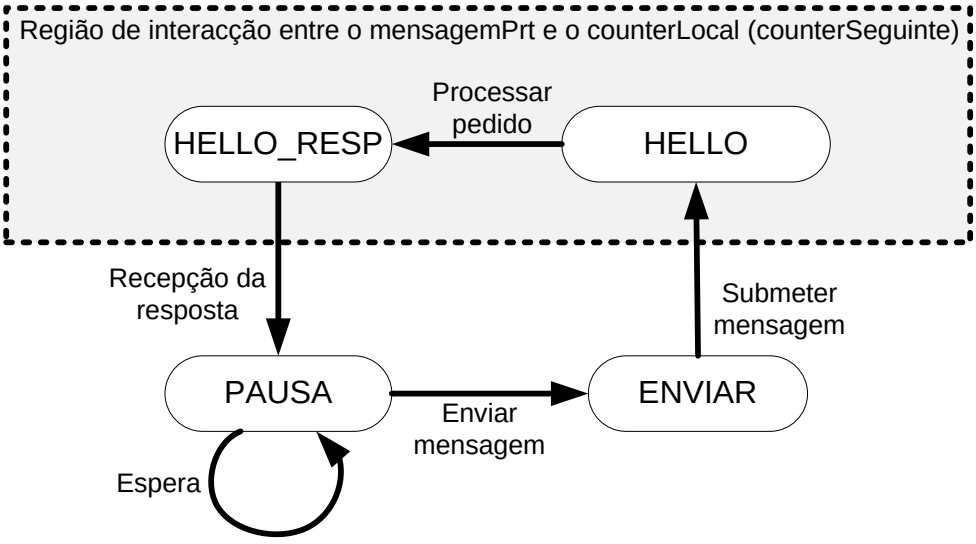

Figura 40 – Ciclo de vida da thread mensagemPrt, responsável pelo envio das mensagens e manutenção do anel

7.3. Descrição das funcionalidades implementadas

A modelação da aplicação foi conseguida através da identificação das diferentes funcionalidades inerentes aos vários protocolos constituintes da arquitectura proposta. Assim, a cada função irá corresponder uma *thread* durante o processo de execução.

Deste modo, tendo em consideração os diagramas de objectos apresentados anteriormente, pode-se verificar que o objecto anel é responsável pela instanciação da estrutura de dados Counter e pela instanciação e lançamento das *threads* joinPrt e mensagemPrt, responsáveis pela execução, como clientes, dos protocolos de inserção no anel e de transmissão de mensagens com verificação do seguinte (protocolo Hello) respectivamente. Ao mesmo tempo é também lançada a *thread* counterLocal, a qual é responsável pela gestão dos pedidos recebidos pelo objecto remoto. A estrutura de dados counter, representando o objecto remoto, implementa o servidor para os protocolos referidos acima, permitindo a recepção de chamadas remotas através da disponibilização de interfaces que aceitam os pedidos provenientes dos Counters anterior e seguinte.

Assim, são definidas três interfaces distintas para este sistema, sendo no entanto todas implementadas pelo mesmo objecto remoto. Cada uma destas interfaces disponibiliza o acesso aos seguintes métodos remotos:

- **counterAnterior** – fornece o acesso à invocação do método remoto que permite o envio de pedidos para inserção no anel (*joinRequest*);
- **counterSeguinte** (outroCounter) – tem como função disponibilizar as funcionalidades requeridas ao nó seguinte no anel, nomeadamente envio de mensagens, com confirmação do Counter seguinte, através do método *sendMsg* e comunicação da ocorrência de falha no anel através do método *ringFail*;
- **submissionCounter** – disponibiliza à aplicação cliente do votante o método que permite a submissão de votos.

Em virtude da necessidade de proceder à devolução do recibo à aplicação votante, esta deverá implementar do mesmo modo uma interface que permita ao Counter seleccionado, para desempenhar o papel de Receipt Counter, a entrega do respectivo

recibo. Nesta situação, caso ocorra uma falha na invocação do método remoto não existe nenhum mecanismo de suporte, sendo o recibo descartado e consequentemente o votante deverá submeter um novo voto para poder obter a confirmação relativa ao sucesso no processo de submissão do voto.

As informações relativas aos Counters e ao anel encontram-se em tabelas armazenadas em ficheiros XML, como já tinha sido referido. A obtenção das informações necessárias para a execução das mais variadas tarefas no anel é conseguida através da estrutura lerAnelXML, a qual fornece métodos para todas as pesquisas sobre as tabelas existentes. Entre os métodos implementados por esta estrutura, destacam-se os seguintes:

- **ObterIntervalo** – possibilita ao Counter conhecer período temporal em vigor e o tempo restante para a próxima alteração da configuração;
- **VerPosicao** – o qual permite a um dado Counter obter a sua posição no anel para o presente período temporal;
- **ObterCounterID** – com este método possibilita-se ao Counter a obtenção do identificador de outro Counter apenas com o fornecimento da posição relativa do mesmo;
- **ObterEndCounter** – através do identificador do Counter é possível obter as informações relativas à sua localização, respectivamente o seu endereço IP e porto TCP.

O sistema desenvolvido executa um número não determinado de iterações, terminando apenas na ocorrência de uma falha ou o cancelamento do funcionamento por parte do utilizador.

7.4. Estrutura das mensagens

Seguindo a modelação apresentada no capítulo 6, foi desenvolvida uma estrutura de dados para a troca de mensagens. A estrutura de dados Mensagem é composta por três atributos, correspondendo dois deles a estruturas em *array*. A definição destes atributos

como *arrays* prende-se com as necessidades constantes de efectuar o *padding* da mensagem em circulação, como será apresentado mais à frente.

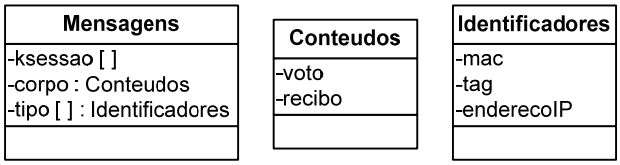

Figura 41 – Definição da estrutura de dados para transporte da mensagem

A Figura 41 apresenta a estrutura de dados definida para o transporte da informação em circulação pelo anel. Assim, para a classe Mensagens são definidos os seguintes atributos:

- **ksessao** – indicador do próximo destinatário da mensagem. Este valor é armazenado num array, sendo a sua posição construído correspondente à ordem de recepção da mensagem. Por outro lado, cada item será cifrado por camadas com as chaves públicas dos Counters até ao seu destino final, mas pela ordem inversa de propagação.
- **corpo** – representa o conteúdo efectivo da mensagem. Definido pela estrutura com o nome Conteudos, possui dois atributos para albergarem o voto e respectivo recibo. A cifra a que estes estão sujeitos deverá seguir o modelo definido anteriormente sobre pena de inutilizar o voto para a eleição.
- **tipo** – este atributo composto, serve para validação e identificação do tipo de mensagem. Deste modo, o atributo mac representa o hash sobre o respectivo ksessao inicial (antes das cifras), podendo o Counter através de comparação determinar se a mensagem se destina a si ou não. Mais uma vez, a posição ocupada por cada item no array corresponde à ordem de processamento da mensagem no anel, sendo cada entrada cifrada do mesmo modo que o descrito para o atributo ksessao. O identificador do tipo de mensagem será guardado no atributo tag, indicando quais as operações a realizar sobre a mensagem. Apenas no caso do *Receipt Counter* o atributo enderecoIP possui informação significativa, representando o endereço para o qual o recibo deverá ser devolvido.

Para a estrutura definida, a operação de *padding*, durante o processamento e transmissão da mensagem, pode ser efectuada através da execução de três passos:

1. Remoção dos itens na posição 0 (topo) dos arrays “ksessao” e “tipo”;
2. Deslocamento de uma posição para cima, para os restantes itens em cada array;
3. Inserção de informação aleatória nos itens que ocupam as posições n-1, considerando que o sistema permite no máximo n processamentos das mensagens.

7.5. Construção do canal seguro

A negociação de chaves de sessão com autenticação tinha sido definida para ocorrer transparentemente antes da primeira troca de dados via RMI entre Counters. Nesta implementação, pretendia-se que o RMI usasse uma classe diferente de *sockets* que cifrassem/decifrassem todo o seu tráfego com uma chave de sessão e com DES em modo CFB8, chave essa negociada através do protocolo de Diffie-Hellman com autenticação. Este protocolo deveria ocorrer em cada ligação TCP inerente a uma interacção via RMI e deveria propagar às aplicações a identidade do interlocutor (quando operando como servidor), ou assegurar uma determinada identidade do interlocutor (quando operando como cliente).

Kay [13] descreve de uma forma sucinta o modelo de programação do RMI, o qual irá ajudar na compreensão dos objectivos propostos. Como se pode observar na Figura 42, o servidor remoto é constituído por dois componentes fundamentais – a interface remota onde são definidos os métodos remotos e a classe onde os mesmos são implementados, a qual deverá estender outras classes de modo a garantir a correcta implementação. De uma forma simplista o mecanismo de comunicação implementado pelo RMI pode ser visto como sistema composto por um *proxy* e um serviço de *backend*.

Para a criação do servidor remoto pode-se optar por dois métodos distintos, *UnicastRemoteServer* ou *Activatable*, correspondendo a servidores sempre activos ou a servidores activos por pedido, respectivamente.

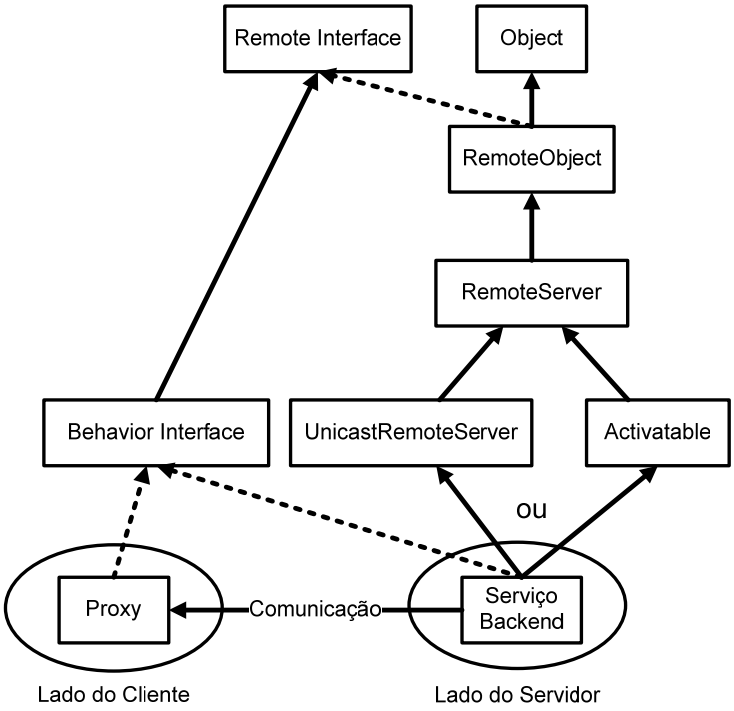

Figura 42 - Modelo de programação RMI

Na arquitectura RMI o *Proxy* do lado do cliente é implementado pelo *Stub*, enquanto que o serviço de *Backend* do lado do servidor é composto pelo *skeleton* gerado pelo RMI e a respectiva classe que implementa os métodos remotos. Deste modo, o conjunto Proxy/Backend encapsulam toda a complexidade das operações de rede, permitindo ao cliente invocar métodos remotos como se fossem locais.

No entanto, no que concerne à adopção do RMI como modelo de comunicação e tendo em conta o seu principal objectivo – tornar a invocação de métodos remotos transparente para o cliente – este revelou-se suficientemente inflexível a alterações significativas da sua estrutura de comunicação (ver Figura 43).

A comunicação entre o cliente e o servidor é assegurada pelas entidades *Sutb* e *Skeleton*. Estas entidades podem ser vistas como os extremos do canal de comunicação cabendo às mesmas a respectiva gestão e ocultando do cliente os detalhes de comunicação. Desde a versão java 1.2 que todos os *Skeletons* dos servidores locais podem ser substituídos por um rmid (RMI *daemon*). Este serviço fica assim responsável pela comunicação com os *Stubs* remotos dos servidores RMI a correrem na máquina local.

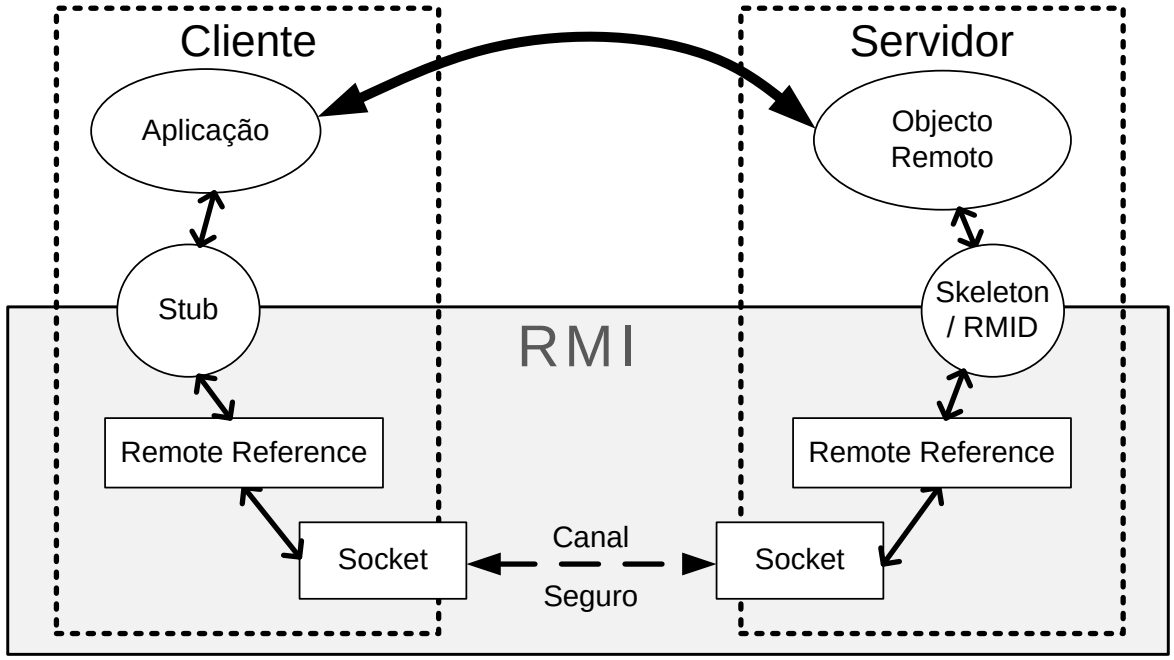

Figura 43 – Arquitectura do RMI com identificação do canal seguro autenticado

Como se pode observar na figura anterior, um sistema RMI é composto por 3 camadas, sendo que cada uma delas interage com a camada inferior. Estas camadas, identificadas em [13], são as apresentadas em seguida e acompanhadas da sua funcionalidade específica:

- Camada *Stub*/*Skeleton* – efectua operações de m*arshalling* / *unmarshalling* relativamente aos argumentos do método remoto invocado;
- Camada *Remote Reference* – responsável pela gestão do tipo de comunicação (*unicast* ou *multicast*) baseada no número de instâncias do servidor, bem como suporte a referências persistentes para objectos remotos independentemente do tipo de activação dos mesmos;
- Camada de Transporte – nesta camada é delegada a responsabilidade pelo estabelecimento e gestão do canal de comunicação, para além das tarefas de seguimento do objecto remoto e encaminhamento de mensagem para o mesmo,

Tendo em conta a arquitectura RMI apresentada acima e baseando a abordagem na documentação disponibilizada pela Sun para a personalização dos *socket factories* em RMI

[14], foi possível desta forma a troca dos valores públicos e consequentemente a construção do canal seguro. Contudo, um dos requisitos impostos aquando da definição do problema correspondia à autenticação mútua dos intervenientes na construção do canal, de forma a evitar ataques por parte de entidades maliciosas. Efectivamente, surge aqui a primeira barreira. Tal como foi apresentado anteriormente, a comunicação num sistema suportado por RMI é estabelecida, transparentemente, entre o *Stub* e o *Skeleton* do objecto remoto, ou seja a gestão da comunicação é exclusivamente da responsabilidade do servidor remoto, não necessitando da intervenção do cliente. Deste modo, uma das falhas detectadas nesta abordagem assenta no facto de o cliente não ter acesso ao *ClientSocketFactory* e consequentemente não lhe ser possível autenticar o seu valor público através da cifra com a sua chave privada nem verificar a identidade do servidor remoto.

Deduz-se assim que, para a introdução da cifra com a chave privada dos valores públicos no algoritmo de Diffie-Hellman, necessitar-se-á de aceder à construção e gestão do *socket* do lado do cliente.

Deste modo, tendo como principal objectivo a construção de um canal de comunicação seguro, existem algumas condicionantes fruto da utilização do modelo de comunicação baseado em RMI. Estes problemas podem ser resumidos do seguinte modo:

1. A instanciação do *stub* não é parametrizável, ou seja, não é possível aquando da instanciação do *stub* (usando a função `registry.lookup`) indicar um parâmetro para a mesma (neste caso poderia ser, por exemplo, a chave pública do Counter que deverá ser “dono” do objecto remoto, para validação da identidade).
2. Não existe qualquer mecanismo base de invocação dos objectos referidos internamente pelo *stub*. Caso tal existisse, seria possível aceder, após a instanciação do *stub*, aos valores públicos de Diffie-Helmann trocados e guardados internamente.
3. A gestão dos *sockets* entre um *stub* e o respectivo objecto remoto é completamente transparente para os programadores, mesmo quando se faz uso das classes de *socket factories*. Por este facto, não é possível saber em que instantes as ligações TCP são criadas, logo não é possível saber os instantes em que deverá ocorrer a troca de valores Diffie-Helmann cifrados.

Portanto, se a mesma acontecer será apenas antes da primeira troca de dados, a qual pode ser detectada, mas que é independente do estabelecimento da ligação TCP por onde os dados serão enviados.

Devido a estas limitações, não foi possível cumprir o protocolo previamente indicado, mas apenas usar trocas de valores de Diffie-Helmann anónimas entre *stubs* e objecto remotos a partir das quais são derivadas as chaves de cifra, que servirão em seguida para cifrar o canal de comunicação (com DES CFB8). Tal funciona mesmo que se criem diversos *sockets* para a mesma interacção *stub*-objecto remoto, mas sempre sem qualquer autenticação.

7.6. Validação e teste

O modelo construído foi testado e validado. Para tal, foram realizados alguns testes sobre a plataforma desenvolvida. A validação da arquitectura foi realizada por fases, sendo criados cenários de funcionamento consoante o estado de operação do anel. Nos vários cenários testou-se a formação inicial do anel, o seu funcionamento em termos de circulação de mensagens, a reordenação do anel em caso de falha temporária ou permanente de um Counter e a reestruturação completa do anel após um intervalo temporal.

- Cenário 1: todos os Counters autorizados estão activos. Este cenário corresponde ao mais simples, uma vez que o anel encontra-se nas condições ideais, sem qualquer falha.
- Cenário 2: falha permanente de um ou mais Counters. Neste cenário consideraram-se situações extremas, em que falham todos os Counters à excepção de dois deles, podendo os mesmos encontrar-se em posições consecutivas no anel ou não.
- Cenário 3: tentativa de uso de um Counter não autorizado. Neste cenário apenas se testa a sua inclusão no anel; os outros testes não podem ter lugar porque a inserção não lhe é facultada.
- Cenário 4: falha temporária num dos Counters integrados no anel, seguida da sua recuperação e consequente tentativa de reinserção no anel.

- Cenário 5: este cenário é semelhante ao anterior mas entre a ocorrência da falha e a recuperação do Counter não decorre tempo suficiente para a detecção da falha pelo Counter na posição anterior.
- Cenário 6: falha em dois Counters consecutivos num anel em completa operação. Para um anel com N Counters, este cenário foi extrapolado para a falha simultânea de N-2 Counters consecutivos.
- Cenário 7: construção do anel fora dos períodos definidos pela eleição. Não se consegue analisar mais nada em virtude de os Counters permanecerem em modo de espera até ao início do primeiro período definido.
- Cenário 8: verificação da alteração da ordenação dos Counters e consequente reconstrução do anel devido ao fim do período temporal definido para a configuração actual.

Este conjunto de cenários de operação permitiu detectar e corrigir erros na especificação e realização dos diversos protocolos de gestão do anel. Neste momento, e com os cenários de teste acima referidos, o sistema permite uma operação constante, as falhas apenas se traduzem na perda das mensagens em trânsito armazenadas nos Counters quando eles falham ou na incapacidade temporária, ou definitiva, de processar um voto (quando o Counter de um LMR está em falta).

Capítulo 8

Conclusões

O objectivo principal desta dissertação prendia-se com a definição do modelo e respectiva implementação de uma nova arquitectura para submissão anónima de votos baseada em Mix Rings. Embora todo o sistema tenha sido desenhado tendo em consideração o sistema de votação electrónica REVS, considera-se que a mesma possa facilmente ser transposta para outros sistemas que necessitem da existência de um canal anónimo para a comunicação entre dois pontos, nomeadamente outros sistemas de votação electrónica, sendo único requisito nestes casos que o novo sistema seja capaz de identificar a existência de cópias, aquando da contagem final, dos votos submetidos.

Sendo a disponibilidade dos sistemas de votação electrónica um factor crucial, esta arquitectura teria de possuir robustez no que respeita a tolerância a falhas ao nível do canal anónimo. Assim, mesmo na eventualidade de ocorrer uma falha em um ou mais Counters, o sistema deveria conseguir recuperar e consequentemente permitir a continuidade de operação do sistema global. Contudo, uma vez que a mensagem com o voto é construída recorrendo à cifra com as chaves públicas dos Counters escolhidos para o seu LMR, torna-se impossível garantir que o sistema recupere de uma falha quando a mesma ocorrer num dos Counters pertencentes ao LMR.

Adoptou-se uma arquitectura com gestão descentralizada para o anel, não existe a figura do coordenador central, sendo da responsabilidade de cada membro a sua inserção e monitorização do estado do seu nó seguinte. Uma vez que a comunicação é unidireccional não existe necessidade de verificação do estado do seu nó anterior. No entanto, para optimização do desempenho e eficiência do sistema, foi desenvolvida uma solução que permite a cada Counter, num dado momento, obter uma aproximação do estado dos restantes Counters à excepção dos seus vizinhos directos, dos quais conhece concretamente o seu estado. O conhecimento aproximado do estado do anel, deve-se essencialmente ao facto de cada mensagem de *cover traffic* ser gerada contendo um Counter intermédio escolhido aleatoriamente, impossibilitando deste modo garantir uma validação periódica de cada Counter. Assume-se que poderão existir atrasos na actualização do estado de operação dos Counters.

Após a iniciação do protocolo de criação do anel verifica-se que o mesmo converge e tende para a configuração definida inicialmente pelo comissário da eleição. Deste modo, considera-se que a convergência do protocolo em conjunto com a variação da ordenação do anel por intervalos temporais, permite que o protocolo proposto usufrua de uma relativa robustez em relação à existência de Counters com comportamentos maliciosos, por exemplo na tentativa de exclusão de um determinado Counter do anel.

Nesta arquitectura, é inviabilizada a participação de Counters não autorizados, o que é conseguido através do controlo da atribuição dos identificadores para cada Counter. Para tal, propõe-se a utilização de um *hash* da chave pública de cada Counter como o seu identificador. Deste modo, garante-se que apenas os Counters autorizados possam participar no anel.

8.1. Trabalho futuro

Como se viu, encontraram-se problemas na utilização do Java RMI no processo de construção do canal seguro baseado no protocolo modificado de Diffie-hellman com suporte para autenticação mútua. Esta funcionalidade deverá ser revista através da procura de alternativas para a constituição deste canal. Estas soluções poderão passar pelo estudo de outras soluções de estabelecimento do canal seguro ou pela definição de modelos diferentes para a comunicação no sistema.

Actualmente a solução apresentada é tolerante a falhas apenas ao nível de operação do RMR. Assim, e tendo em conta o requisito de tolerância a falha imposto ao sistema, uma das propostas de trabalho futuro que se impõe fazer passará pelo estudo de soluções que permitam a introdução do suporte para tolerância a falhas no LMR. Esta solução poderá passar pela adopção de mecanismos semelhantes aos descritos no artigo *Fault Tolerant Anonymous Channel* [15].

De modo a incrementar a eficiência global do sistema, propõe-se que sejam estudados mecanismos que, em tempo real, sejam capazes de analisar e avaliar os tempos limite para expiração da validade das mensagens contendo votos, passando essa informação ao cliente no momento de submissão. Deste modo, seria possível ao votante identificar o momento a partir do qual deveria efectuar uma nova submissão no caso de não ter recebido a confirmação de submissão efectuada com sucesso. Esta validade estaria intrinsecamente ligada às condições de sobrecarga da rede e taxa de utilização do sistema.

Tendo em consideração o sistema de verificação do estado dos Counters presentes no anel, seria interessante o estudo de soluções para a geração de mensagens de *cover traffic* baseadas em mecanismos pseudo-aleatórios para a escolha dos Counters intermédios. Com a introdução destes mecanismos poderia ser possível delimitar os tempos máximos de verificação do estado de cada Counter e, assim, fornecer informações mais precisas às aplicações cliente.

Um dos pressupostos definidos aquando do desenvolvimento desta arquitectura centrava-se na concordância de que nenhum dos Counters produziria ataques do tipo DoS. Assim, de forma a incrementar a robustez do sistema no que toca a ataques maliciosos por parte dos intervenientes, seria interessante o estudo e definição de mecanismos que previnam contra este tipo de ataques ou mesmo que tolerem a existência de Counters bizantinos.

Glossário

REVS Robust Electronic Voting System

P2P Peer-to-peer

ALM Application Layer Multicast

PSF Ponto Singular de Falha

RMR Real Mix Ring

LMR Logical Mix Ring

TCP Transport Control Protocol

RMI Remote Method Invocation

RP Rendevouz Point

VRR Virtual Ring Routing

DHT Distributed Hash Tables

SSSR Self-Stabilizing Structured Ring

DoS Denial of Service (Negação da prestação de serviço)

Bibliografia

[1] A. Juels and M. Jakobsson. Coercion-resistant electronic elections. Cryptology ePrint Archive, Report 2002/165, 2002.

[2] M. Burnside and A. D. Keromytis. Low Latency Anonymity with Mix Rings. In 9th Information Security Conf. (ISC 2006) (LNCS 4176), pages 32–45, Samos, Greece, Agosto 2006.

[3] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Comm. of the ACM, 24(2):84–88, 1981

[4] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous Connections and Onion Routing. In 18th IEEE Symp. on Security and Privacy, pages 44–54, Oakland, CA, USA, Maio 1997.

[5] André Zúquete e Filipe Almeida. Verifiable Anonymous Vote Submission. ACM Symposium on Applied Computing (SAC 2008), pages 2159-2166, Fortaleza, Brasil, Março 16-20, 2008.

[6] R. Joaquim, A. Zúquete, and P. Ferreira. REVS – A Robust Electronic Voting System. IADIS Int. Journal on WWW / Internet, 1(2):47–63, Dezembro. 2003.

[7] Filipe Almeida e André Zúquete. Mix Rings Tolerantes a Falhas para Submissão Anónima de Votos. 3ª Conferência Nacional sobre Segurança Informática nas Organizações (SINO'2007), Lisboa, Portugal, Novembro 7-8, 2007.

[8] A. Sobeih, W. Yurcik, and J. C. Hou. VRing: A ring-based application-layer multicast protocol. Tech. Rep. UIUCDCS-R-2004-2468, University of Illinois at Urbana-Champaign, Agosto, 2004.

[9] Matthew Caesar, Miguel Castro, Edmund B. Nightingale, Greg O'Shea, Antony Rowstron. Virtual Ring Routing: Network Routing Inspired by DHTs. Sigcomm 2006, Pisa, Italy, Setembro, 2006.

[10] Ayman Shaker and Douglas S. Reeves. Self-Stabilizing Structured Ring Topology P2P Systems. Fifth IEEE International Conference on Peer-to-Peer Computting, pages 39-46, Agosto-Setembro, 2005.

[11] Ronald L. Krutz e Russell Dean Vines. The CISSP Prep Guide: Gold Edition. Wiley Publishing, Inc. 2003

[12] W. Diffie and M.E. Hellman. New directions in cryptography. IEEE Transactions on Information Theory 22 (1976), 644-654.

[13] Kai Tan. Pattern-based Parallel Programming in a Distributed Memory Environment. Universidade de Alberta, 2003.

[14] Sun Microsystems. Using Custom Socket Factories with Java RMI, 2006. http://java.sun.com/javase/6/docs/technotes/guides/rmi/socketfactory/index.html

[15] W. Ogata, K. Kurosawa, K. Sako, and K. Takatani. Fault Tolerant Anonymous Channel. In 1st Int. Conf. in Information and Communications Security (ICICS'97) (LNCS 1334), pages 440–444, Beijing, China, 1997. Springer-Verlag.